Dresden University of Technology
Faculty of Computer Science
Institute of Software and Multimedia Technology
Chair of Software Technology

Diploma Thesis

Clustering of Distributed Word Representations and its Applicability for Enterprise Search

Christina Korger
Born March 16, 1991 in Regensburg
Matriculation Nr. 3703541

Supervision:
Dr. Birgit Demuth TU Dresden
Dr. Uwe Crenze interface projects GmbH
Professor:
Prof. Dr. rer. nat. habil. Uwe Aßmann

Submitted on July 28, 2016

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Software- und Multimediatechnik, Professur Softwaretechnologie

Technische Universität Dresden, 01062 Dresden

Prof. Dr.rer.nat.habil.
Uwe Aßmann

Sekretärin: Katrin Heber
Telefon: 0351 463-38464 oder 38463
Telefax: 0351 463-38459
E-Mail: uwe.assmann@tu-dresden.de

Dresden, 19.02.2016

Aufgabenstellung Diplomarbeit für Christina Korger

Thesis Topic (Thema)

Clustering of Distributed Word Representations and its Applicability for Enterprise Search

Clusterung verteilter Wort-Repräsentationen und dessen Anwendbarkeit für die Unternehmenssuche

Assignment (Aufgabenstellung)

The hypothesis that the meaning of a word can be exhaustively described through its context is commonly attributed to the linguists Harris and Firth and the 1950s. As a logical consequence the similarity of words can be expressed as a function of context similarity. With the help of simplified neural network models modern computers learn a reduced distributed representation of words as vectors in feature space. This group of techniques, also referred to as neural word embeddings, constitutes a popular approach to natural language modelling in state-of-the-art research.
Knowledge management tools for enterprises aim to support their users by preparing and presenting information in a way to help them overcome the otherwise inaccessible amount of accelerating data. With enhanced understanding of semantic relationships, systems become more adaptable to the apparent irregularities of natural language and are enabled to assume even larger responsibilities. Similar to recent developments in web search, they can provide recommendations to refine, complement or disambiguate a query.

A typical use-case from general vocabulary is the word *Erde* which can either refer to the Earth as a planet or earth as soil. Based on semantic understanding, one good alternative for a system to deal with this is to provide results for the word sense which is more probable in the given context and give a link to the other one in the style of “Did you mean…?”. For this intention, however, it is necessary to find a suitable differentiation such as “Erde (Planet)” or “Erde (Boden)”. If aware of the position in a simplified taxonomy, it is furthermore possible to offer UI controls for extending, shifting or refining the search area.
Compared to other approaches, word embeddings hold out the prospect of low manual effort together with good performance and certain customisation options which is specifically attractive for the field of enterprise search. In cooperation with the local software company *interface projects*, the results of subsequent term clustering on German word embeddings shall be explored. Expected outcome is insight into how well these capture the semantic relationships described above. Although recent findings indicate remarkable improvements of distributed models towards more traditional approaches in many tasks, there is scope for further research, especially in respect to the German language and the enterprise search infrastructure.

The assignment includes the following subtasks (Teilaufgaben):

- Literature review of state-of-the-art word embeddings and related language modelling techniques
- Examination of clustering algorithms and their applicability for capturing semantic relationships
- Development of an experimental setup for evaluation of neural embedding models with different parameters
- Development of an experimental setup for evaluation of different term clusterings on these models
- Analysis of obtained results
- Conceptual integration of the investigated capabilities with the search platform *intergator*

Statement of Authorship

I hereby confirm that this diploma thesis titled "Clustering of Distributed Word Representations and its Applicability for Enterprise Search" has been composed by myself and is based on my own work, unless stated otherwise. No other person's work has been used without due acknowledgement in this thesis. All references and verbatim extracts have been quoted, and all sources of information, including graphs and data sets, have been specifically acknowledged.

Dresden, July 28, 2016

Acknowledgement

At this point, I wish to take the chance and express my sincere thanks to those who contributed to the completion of my studies and, primarily, this diploma thesis. I thank Maximilian Köper for some valuable information via email and for making his test collection publicly available, as well as Sabine Schulte im Walde for providing me with the original database of paradigmatic semantic relations. My thanks go to those who accompanied and supported me up to the present day, or gave me the right impulses at the right time. Many thanks to the Chair of Software Technology for the patient audience to my presentations over the last years and valuable feedback. I specifically thank Professor Aßmann and Mrs. Demuth as my supervisor for their openness towards my research proposal and their cooperation with the software company interface projects. I thank my colleagues for their flexibility in collaboration and the shared enthusiasm for an exiting topic. Mr. Crenze as managing director and external supervisor, let me express my particular gratitude for opportunities and freedom in development in a constructive work environment, at technical, but also at a personal level, as well as for his confidence in me and his time for questions and discussion. Many thanks to my close friends, my family, and especially you, Johannes, for your faith in me. You have always been there for me and have given me your love and affirmation, each in your own way.

Contents

1 Introduction

Machine learning of distributed word representations with neural embeddings is a state-of-the-art approach to modelling semantic relationships hidden in natural language. The thesis "Clustering of Distributed Word Representations and its Applicability for Enterprise Search" covers different aspects of how such a model can be applied to knowledge management in enterprises. A review of distributed word representations and related language modelling techniques, combined with an overview of applicable clustering algorithms, constitutes the basis for practical studies. The latter have two goals: firstly, they examine the quality of German embedding models trained with *gensim*[1] and a selected choice of parameter configurations. Secondly, clusterings conducted on the resulting word representations are evaluated against the objective of retrieving immediate semantic relations for a given term. The application of the final results to company-wide knowledge management is subsequently outlined by the example of the platform *intergator*[2] and conceptual extensions.

1.1 Motivation

With increasing amounts of digital data, software support in managing these becomes a vital necessity for enterprises. Previously, employees who needed a computer for their tasks where individually responsible for establishing and maintaining a systematic order within documents and their storage in a hierarchical file system. Due to extended requirements, this has rapidly evolved into an impossible task. One decisive factor is collaboration: the more content is managed electronically, the more collaboration is moved to digital space. Multiple people work on overlapping topics - thus, preserving a consistent structure produces high additional effort. But especially in the first phases of a project it can be important to neglect strict organisation in favour of creativity and ideas. In other cases, it just makes work less efficient having to memorise where to find and where to put respective information. As an additional difficulty, when an old storage system becomes too slow or too small, it rarely gets fully replaced. More often, a new system is installed next to it and only some of the data is transferred. While much of the rest will never be needed again, access has to be assured for limited occasions. Varying use of third-party tools and their diverse storage strategies increases the desire for simplified access, regardless of location, through one interface.

The general solution to the described problem has long proven its worth for the internet where similar conditions apply: search engines. *Google* as acknowledged expert in web search

[1] A Python library for natural language processing [44]
[2] Product of the company *interface projects GmbH*

has dedicated itself to making the former search bar the personal digital assistant of each user with his individual information needs. It facilitates powerful information retrieval, enabling not only to search for any type of content but to find answers to questions which could not even be formulated beforehand. Library management on the other hand has a long history in systematic cataloguing by title, author, keywords and other meta information. Many approaches of both web search and library management have been adopted for information retrieval on corporate data, described respectively by the terms enterprise search and knowledge management. Solutions such as the platform *intergator* offer flexible access to relevant resources together with possibilities to customise information retrieval and preparation. Provided a rich presentation of results, information can be explored both textually and visually. Example product components are short previews such as images or text snippets, dynamically generated to contain potentially relevant information, associated persons and category information. Based on the initial response, users are offered multiple options to sort and filter, as well as to refine or extend their original query. With more and more data created due to progressive digitalisation, it is essential for information retrieval systems to remain open for improvement and completion.

Great potential for exploratory search via refinement or extension lies in neural word embeddings which have recently proven a serious competitor in the field of natural language processing [4, 28, 31, 32]. Rule-based approaches to text analysis have the drawback that, by triggering uncovered exceptions of the implemented rules, users experience unexpected behavior. The implementation has no understanding of the actual relations between words. One possible consequence is to refine and add additional rules. The German language, however, provides numerous of these exceptions and history shows that language is subject to continuous evolution. Traditional n-gram models in comparison do not capture a semantic representation of words, which can be seen as the elemental building blocks of meaning in language, but mere sequence probabilities. Complying with the principle "you do not understand what you cannot explain", a deeper understanding is necessary to truly identify relationships between words. Hypernyms and hyponyms, generic concepts and their more specific forms, are valuable tools also in professional discussions to achieve transparency and understanding. In 2013, Google published a collection of algorithms and techniques with the name *word2vec*. Although based on well-known techniques, word2vec has had great success in making language modelling accessible to a wide public and adaptable to its application context via several hyperparameters. Common to all possible choices is a simplified neural network learning model and resulting dense representations of each word's natural context as multi-dimensional feature vectors.

The challenge with vector spaces of high dimensionality is to port the encoded information back to a human-understandable form. One viable solution is to conduct some data analysis in the background and support the user with automatic (textual) suggestions based on semantic relationships. As a second solution, additional user interface components can be added which enable dynamic exploration and visualisation of results. For both, it is essential to identify approaches suitable to the use case among the wide range of algorithms devoted to classification, clustering and dimensionality reduction. The objective always is to reduce information of a certain complexity to a coarser division observable from a certain perspective. Classification in machine learning is distinguished from clustering as matching entities to a given set of classes. The aim of clustering algorithms, also referred to as unsupervised classification, is to automatically determine a good partitioning of the data that results in groups with high

inner coherence and low coherence between each other. For high-dimensional data it can be necessary to apply dimensionality reduction in the form of feature selection or transformation, either leading to better clustering results when used in a preliminary step, or directly and facilitating visualisation of partial aspects.

While much research on distributed word representations has been published over the last years, it is very difficult to obtain an overall picture of crucial factors for training and clustering of distributed word representations. The reference to statistical and linguistic foundations is frequently concealed by complexity or inconsistent terminology. Additional shortcomings reside in statistical significance of reported results where, even for exhaustive analyses, often a partial omission of restricting factors and parameters can be observed. Furthermore, there are various efforts towards the urgent need of uniform methods and test collections for evaluation. As English is the universal and academic language, much research focuses on investigations with English corpora and language models. However, the language imposes only a small subset of the issues which occur with morphologically rich languages such as German. Finally, the missing relation to task-specific application is frequently criticised. The company interface projects has particular interest in distributed word representations and their potential to improve enterprise search, especially from a domain- and company-specific perspective. The medium-sized Dresden software company has many German but also international customers for whom English is not the primary document language. With this thesis, the scientific foundations shall be provided and several of the claimed capabilities shall be reviewed.

1.2 Thesis Structure

The next chapter is dedicated to related work, including noteworthy publications in the field of interest as well as the most influential literature for the following chapters. Chapter 3 offers an introduction to distributed language models. Here, terminology and concepts of word embeddings, distributed representations and neural networks are brought into a common context. Chapter 4 provides a systematic outline of popular clustering techniques. Chapters 5 and 6 document the experimental setup. While the first concentrates on evaluation of neural embedding models, trained with different preprocessing steps and training parameters, chapter 6 covers subsequent term clusterings on these distributed representations with the objective of unsupervised retrieval of hierarchical semantic relationships. Analysis of obtained results is given at the end of each chapter. Chapter 7 complements the work with a conceptual integration of the investigated capabilities with the company's search platform *intergator*. Chapter 8 concludes with summary and discussion of conducted work together with an outlook on open research questions.

2 Related Work

The idea that language can be modelled as a pattern of word co-occurrences and therefore, each word can be described by its context, was promoted by Harris [16] and Firth [11] in the 1950s. Regardless of restricted capabilities for technical application, their theoretical analysis of distributional structure in language and subsequent considerations pioneered and inspired scientific research in machine learning of natural language. The stated descriptive expressiveness of a word's context is well known as the *distributional hypothesis*.

The beginnings of artificial neural networks as a means to facilitate machine learning reach back to a similar period. In 1949 psychologist Donald O. Hebb had published his theory on associative human learning through reverberatory activity of nervous cells [18], known today as *Hebbian theory* or *Hebb's rule*: It says that one cell, repeatedly participating in excitation of another, establishes or increases the stability of a connection to it. This theory and its promise of natural robustness against errors, yet great expressiveness, created the idea to emulate neural network learning with computers. One milestone constitutes the *perceptron convergence theorem* [46] by Frank Rosenblatt about one of the first artificial neural network algorithms and its convergence properties, although criticised by Minsky and Papert [34] for not being able to learn non-linearly separable classification functions such as the logical XOR. Several years later, the proposal of multi-layer perceptrons and *backpropagation learning* revived interest in neural network research. Despite slower convergence and the chance of the underlying gradient descent getting caught in local optima, it resolved the former limitation in that it could model arbitrary problems. An article published in the international journal *Nature* by Rumelhart et al. [48] details the incremental process of learning internal representations in neural networks with backpropagation and stochastic gradient descent. Finally, the two volumes on "Parallel Distributed Processing" [50, 51], a collaborative work by Rumelhart, McClelland and the PDP Research Group, document further investigation of the parallels between distributed learning of the human brain and the distributional structure of natural language.

With a significant increase in processing power, unsupervised learning with artificial neural networks has regained interest in multiple research areas related to computer science. Foundations of state-of-the-art word embeddings include the language models introduced in [5, 39] and [35]. Bengio et al. [5] present a probabilitstic feedforward neural network which learns distributed word feature vectors as a cure to problems of data sparsity and high dimensionality with sequential models. Morin and Bengio [39], as well as Mnih and Hinton [35] follow the approach of reducing both training and test complexity with a hierarchical neural language model. By organising words in a tree structure, they enable a hierarchical decomposition of word probabilities with logarithmic complexity. In [56], Turney and Pantel

present a systematic literature review of state-of-the-art *vector space models* (VSMs) at that time and their applications to semantic tasks. They classify vector space models by their underlying matrices, thereby emphasising the relationship between VSMs based on a term-document matrix as underlying common document ranking functions and other types such as distributed word representations, derived from a word-context matrix.

In 2013, a toolkit named *word2vec*[1] was released along with several publications [31–33, 36], especially "Efficient Estimation of Word Representations in Vector Space" [31] and "Distributed Representations of Words and Phrases and their Compositionality" [32], where Mikolov et al. introduce the two underlying neural network model architectures *Continuous Bag-of-Words* and *Skip-gram* and two different learning objectives, along with additional parameters for individual optimisation. The first learning objective can be formulated as a minimisation of the error between target and computed output of a forward pass through the neural net, which is closely related to *backpropagation of errors* as detailed in [48], with the advantage of using *hierarchical softmax* [35, 39] based on a binary Huffman tree for normalisation of probabilities. The second objective is a maximisation of the corpus probability in a contrastive approach called *negative sampling*, derived from noise-contrastive estimation as introduced by Gutmann and Hyvärinen [14] and Mnih and Teh [37], for the application to neural word embeddings. Valuable insights into the word2vec toolkit originate, amongst others, from Levy and Goldberg [12, 25–27] as well as Levy et al. [28]. The authors provide a different explanatory approach for some of the algorithms, explore and compare performance of different parameter configurations and evaluate against related models. In [25] Levy and Goldberg explore word representations trained with dependency-based contexts. Instead of the word2vec default linear context sampling, they construct each word contexts based on preliminary syntactic dependency parsing, producing interesting differences such as a preference of functional similarities towards domain similarities. The qualitative and quantitative evaluation offers additional insight into how the choice of context decides the resulting models' capabilities. In [26], the authors analyse the relationship between word2vec embeddings and explicit word-context VSMs, as well as the vector arithmetic behind automatically solving word-analogy tasks. They identify the originally proposed method as a linear combination of pairwise similarities and introduce a modified version, referenced and, meanwhile, widely adopted as *3CosMul*. Finally, in [27], Levy and Goldberg compare word2vec in the configuration skip-gram and negative sampling to alternative word representations based on *pointwise mutual information* (PMI).

GloVe[2], a model yielding similar distributed word vector representations, was developed at Stanford by Pennington et al. [42] and is part of a number of comparisons. Baroni et al. [4] evaluate word2vec CBOW models in several configurations in comparison to statistical models with and without reduced context space. Their findings indicate a considerable advantage of CBOW, both across popular benchmarks and in terms of training time.

The established standard work on "Foundations of Statistical Natural Language Processing" by Manning and Schütze [29] provides further proof of the existence of many of the algorithms and mechanisms contained in word2vec long before hardware capabilities and the amounts of electronic data enabled utilisation for knowledge access of the current magnitude.

[1] `https://code.google.com/archive/p/word2vec/` [accessed 6 June 2016]
[2] `http://nlp.stanford.edu/projects/glove/` [accessed 6 June 2016]

Rei and Briscoe [45] take an approach at hyponym generation using different vector space models and similarity measures. To retrieve subcategories of general terms like 'sport' or 'treatment', they introduce a weighted cosine similarity measure and evaluate several models, among them two skip-gram models and a dependency-based model, which achieves the best overall results. Köper et al. [22] analyse to which degree distributed word representations trained with word2vec preserve more difficult relationships such as paradigmatic relations. In their test series they report results for English and German corpora, both with and without preliminary lemmatisation.

In order to evaluate different solution approaches for the problem at hand, benchmark tests are frequently utilised. Thereby, results are always dependent on both the model under assessment and the criteria of the test. Various work is concerned with developing datasets for evaluation purposes. It can generally be distinguished between three types of test collections: Those concerned with evaluating the representation of *semantic similarity* between two words, collections for evaluation of *semantic regularities* between word pairs through analogy tasks and data sets for task-specific evaluation, e.g. tagged corpora for evaluation of *named entitiy recognition*. Well established publications with repeated use of their test collections are [10,47] as well as [24]. Rubenstein and Goodenough [47] provide 65, Finkelstein et al. [10] a collection of 353 English word pairs with human-annotated similarity scores. The latter is further divided by Aggire et al. [1] into two separate sets, differenciating between actual semantic similarity and mere topical relatedness. Landauer and Dumais [24] use a set of 80 multiple-choice questions from the official *Test of English as a Foreign Language* (TOEFL) to evaluate their proposed unsupervised learning model, known by the name *latent semantic analysis* (LSA). Additional to empirical evaluation, their work comprises an extensive reflection of how and in which respects mathematical computation can model human knowledge acquisition. Mikolov et al. publish a test collection of 8 000 analogy pairs in [33], further extended to 9 128 pairs with focus on non-syntactical relationships in [31].

Although less scientific research concentrates on construction of German language test sets, for the collections mentioned above, respective counterparts have been created. In [13], Gurevych documents the translation of Rubenstein and Goodenough's 65 similarity pairs [47]. Schmidt et al. [53] create an equivalent to the 353 word pairs in [10], reduced to 280 pairs with regard to conceptual transferability. Mohammad et al. [38], similar to [24], propose measuring correlation of language models to human intuition with a set of 1008 multiple choice questions. The questions are extracted from several German issues of the magazine *Reader's Digest* with the task to correctly select the one out of four alternatives most closely related to a given target. Köper et al. [22] construct a test set of 18 552 analogy pairs as the German counterpart to the one with syntactical and semantical analogy pairs in [31] with omission of the relation-type "adjective-adverb"[3], as well as an additional test set of 1 684 analogy pairs from a database collected by Scheible et al. [52]. The latter is composed of a representative distribution of terms from GermaNet, categorised by word class and the three relation types antonymy, hyponymy and synonymy, i.e. target - opposite, target - broader term and target - synonym. Recent work [9,54] focuses on the quality and significance of evaluation techniques. Schnabel et al. [54] make four specific contributions: They analyse the relationship of different evaluation criteria, propose a concept of how to collect

[3]for the majority of pairs, German adjective and adverb are identical

human annotations through crowdsourcing, demonstrate how to construct a representative test set and trace back partial variability across embedding models and evaluations to interfering frequency effects. Faruqui et al. [9] equally emphasise the lack of standardised methods for evaluation in respect to specific tasks and express their concerns about drawing incorrect conclusions from unstable results. In [58], Wilson and Schakel introduce the insertion of artificial tokens to a natural language corpus for a more controlled evaluation of word embeddings.

A general overview of clustering algorithms from the natural language processing perspective is provided by Manning and Schütze [29]. In their well-known standard work on "Applied Multivariate Statistical Analysis", Härdle and Simar [15] offer a different view on clustering with focus on applicable visualisation techniques for high-dimensional data analysis. The problems and capabilities of cluster analysis for high-dimensional data are a popular subject of discussion in the field of knowledge discovery and data mining. Parsons et al. [40] offer a survey of various clustering algorithms applying *subspace clustering*.

The literature presented in this chapter is collected from various research areas, just like we defined subtasks for a number of different problems. Starting from the top, in the following chapter, we are going to detail the concepts of neural network learning for natural language modelling.

3 Distributed Word Representations

The *distributional hypothesis* states that the similarity of contexts that words appear in is closely connected to their similarity in meaning. A popular citation that summarises this view is that of J.R. Firth from 1957: "You shall know a word by the company it keeps" [11]. Distributed word representations based on *neural embedding* are a group of algorithms implementing this hypothesis and constitute a state-of-the-art approach to natural language modelling. Neural embedding, in particular, denotes the process of unsupervised learning with the help of artificial neural network structures that yields dense word vectors. In this process, syntactic and semantic information of each word, defined by the distributed contexts it occurs in, gradually becomes encoded within a reduced number of feature dimensions. Once learned, these vectors or *word representations* of a model can be used for an exploration of regularities in the original vocabulary. The first section revises milestones in the history of natural language modelling with focus on neurolinguistics. Thereafter, the model of a biological neuron is outlined, together with a discussion on how the concepts are transferred to artificial neural networks. Notational conventions are defined in the course of the analysis of the toolkit *word2vec* and its combined algorithms, adjustable to a given use case by a number of hyperparameters.

3.1 History

Local representations of words as the counterpart to distributed representations play a major role in the history of natural language modelling. The characterisation 'local' signifies that representations do not express a word as built of different components but as an atomic unit. Simplicity of this kind of representation makes it easy to handle but hard to express any degrees and facets of similarity between words. N-grams as popular member of this group have a long history and are a common technique for estimating the probability of a sequence. Considering an exact sequence, however, poses certain disadvantages. Even a small vocabulary of 1000 words amounts to 1000^4 possible combinations considering (only) four-grams. For an approximation with four-grams and the example $the\,cat\,chases\,the\,mouse$, this means approximating the sequence probability as a Markov Chain of order 3:

$$
\begin{aligned}
P(\text{“the cat chases the mouse”}) = \quad & \prod_{i=1}^{5} P(w_i|w_1...w_{i-1}) \approx \prod_{i=1}^{5} P(w_i|w_{i-3}, w_{i-2}w_{i-1}) \\
= \quad & P(the) \cdot P(cat|the) \cdot P(chases|the, cat) \\
& \cdot P(the|the, cat, chases) \\
& \cdot P(mouse|cat, chases, the)
\end{aligned}
$$

The probability $P(mouse|cat, chases, the)$ does not reveal anything about the semantic proximity of the word $mouse$ to the word cat (nor a single other word), but merely depends on how many times in proportion to other words $mouse$ has been observed following the exact sequence $cat\,chases\,the$. On the one hand, complexity and missing training examples force us to reduce our examination to this small context of three preceding words. Considering a sequence of ten words would mean a high exponential increase in possibilities while making an accurate statement on its probability much less likely due to the insufficient number of observable occurrences during training; syntactically and logically correct combinations might not be part of the training set at all and must be estimated. On the other hand, taking tri- or four-grams there is no possibility to respect key terms outside this small field of vision. For the test sequence $the\,cat\,eagerly\,chases\,the\,mouse$ and only similar occurrences like $the\,cat\,chases\,the\,mouse$ (above) or $the\,cat\,chases\,a\,mouse$ observed during training, the $n-1$ preceding words $eagerly, chases, the$ offer little indication of $mouse$ being an appropriate successor. Far more challenging, there is no relation between the probabilities $p(mouse|cat, chases, the)$ and $p(mouse|eagerly, chases, the)$. The probabilities $p(mouse|cat, chases, the)$ and $p(dog|cat, chases, the)$ at least would only differ in the last factor. Natural language processing has developed several techniques such as removing very frequent stop-words and stemming to improve results, but it still seems it would have a rather poor chance of competing with a human on the task of usefully complementing a sentence or blanks in a text. Despite its flaws, n-grams are still used in many applications - they offer usable results at reasonable cost. However, the present state of hardware and software capabilities encourages reconsideration of this decision. Parts of the construct do not align well with the human intuition of similarity: should there not exist a closer correlation between the examples $the\,cat\,chases\,the\,mouse$, $the\,cat\,chases\,a\,mouse$ and $the\,cat\,eagerly\,chases\,the\,mouse$?

Following the *distributional hypothesis* [11, 16], words can be compared via the context they appear in. A closer correlation between the examples mentioned above is exactly what can be achieved with distributed word representations: the probability of a word can be inspected, independently from rigid word order, semantically interchangeable or meaningless expressions, based on the strength of association to surrounding words. One step in this direction poses the *bag-of-words* model. Representations per word enable a more flexible consideration of surrounding text, firstly, in both directions, and secondly, without specific order.

With this characteristic an additional parallel to the most popular measure used for document retrieval and ranking becomes apparent: term frequency-inverse document frequency or, in short, *tf-idf*. Turney et al. [56] classify three different types of vector space models (VSMs), i.e. the ones based on term-document matrices, pair-pattern matrices and word-context matrices. Tf-idf precisely constitutes a term-document VSM for semantic processing of documents,

which, in the proper sense, would be correctly paraphrased as a model of distributed document representations. Just like with distributed word representations, the order of terms within documents is ignored for tf-idf. Opposite to word-context VSMs, however, this distribution does not extend to words or phrases, atomic carriers of meaning.

The disregard of word order might be one, but certainly not the only condition for the success of distributed language models. Other significant conditions include different optimisations that improve efficiency as far as to facilitate utilisation of the large datasets available nowadays, while taking advantage of any opportunities to reduce computation overhead where little can be gained. The following shall illustrate the ingenious combination of techniques manifesting in *word2vec*, but also that the ideas have long been developed and refined separately, and it might be due to advanced collaboration and intercourse in research that we can create and evaluate complex solutions such as this.

The idea to compare and align machine learning of natural language with neural network-like structures already occurs in the works of Hinton et al. [19]. Artificial neural networks enable unsupervised learning of such a distributed representation, and will be further illustrated in a brief digression.

3.2 Parallels to Biological Neurons

Biological neurons are responsible for processing and transmitting information through our bodies and inside our brain via electrical impulses. To understand the concepts behind artificial neural networks it is useful to recall structure and interaction of a neuron cell.

For the typical structure of a biological neuron, see figure 3.1. Depicted on the left is the cell body or *soma* with the *nucleus* in the center and outgoing ramifications, called *dendrites*. The soma with its dendrites offers the surface for other neurons to transmit signals over their *synaptic terminals*. Over the *axon* extending to the right, the neuron in turn can propagate a signal and excite subsequent neurons. The *myelin sheath* insulates the axon and prevents impairment or loss of the signal on its way to the axon terminals. Whether or not a signal is propagated to neighbouring neurons depends on whether the accumulated inputs exceed an internal action potential.

Similarly, computational models can be constructed to learn distributed representations of words. Namely they resemble artificial constructs in the following aspects:

Incoming Signals The signals received from neighbouring neurons at dendrites and soma are the input values of an artificial neuron. Both can be in the order of thousands.

Input Weights Independent from the transmitted signal strength or the input value itself, where an input neuron's synapse connects to the neuron under observation also affects the received signal. Weighting is also applied to the input of an artificial neuron.

Summation Function Negative and positive ions are accumulated in the cell body, effectively amounting to a summation. Weighted inputs of an artificial neuron are also summarised.

Threshold The electrical potential required to accumulate at the opening of the axon before a signal is eventually propagated forward to the dendrites and somata of subseding neurons

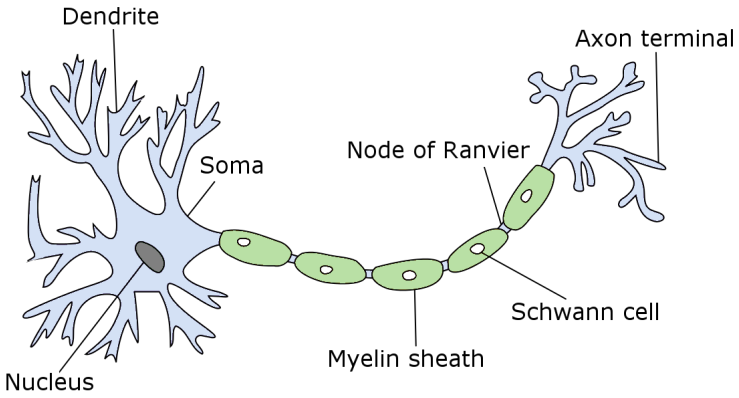


Figure 3.1: Biological Neuron. *The neuron can be excited by signals from preceding neurons transmitted to dendrites and soma. When reaching a certain action potential a signal is propagated to succeding neurons over its axon terminals. Adapted from Wikimedia Commons, by Quasar Jarosz at English Wikipedia, CC-BY-SA-3.0*

is sometimes implemented as the threshold of an artificial neuron.

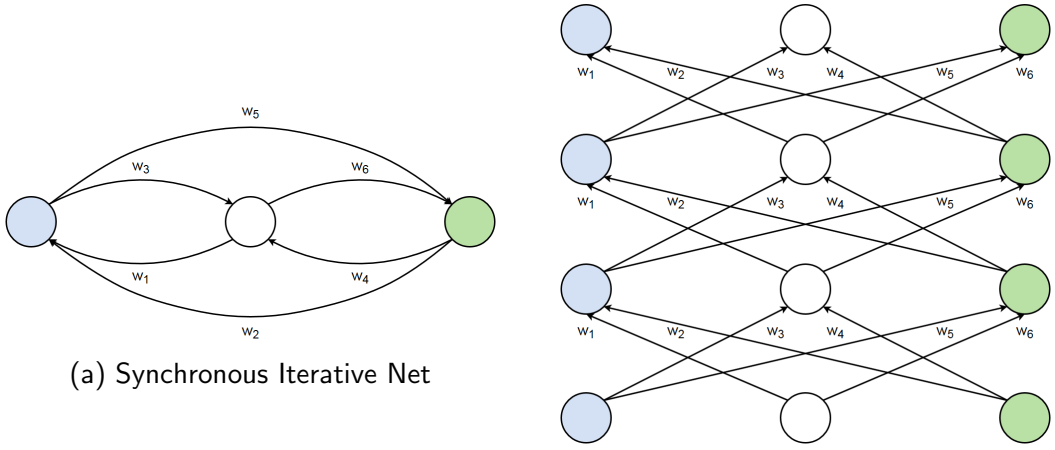
Opposite to supervised learning algorithms there is no handcrafted groundwork to be done, quality of the resulting model is therefore due to the configuration of its training parameters, sufficient size of the training data and compliance between training and application domain. To analyse, empirically evaluate and confirm information available about a use-case-driven parameter choice is the focus of section 3.5 and chapter 5.

3.3 Feedforward and Recurrent Neural Networks

Feedforward neural networks constitute a class of artificial neural network models composed of an *input layer*, an arbitrary number of processing or *hidden layers* and an *output layer*. All connections link between two neurons of adjacent layers and are directed toward the output, which by definition excludes any cycles. The term *recurrent neural networks* (RNN) applies to neural network models with directed cycles. RNNs come closer to the idea of simulating a biological neural network in how they model variance over time - by back-coupling, neural units can mutually influence the states of one another. Networks with each neuron linking to all neurons of the next layer are also referred to as completely linked, or in the case of RNNs, fully recurrent. To enable application of the learning procedure discussed below (see section 3.4) a recurrent network can be mapped to a feedforward network with corresponding weights as shown by Rumelhart et al. [48]. One example of such a mapping is illustrated in figure 3.2.

A feedforward net with corresponding weights preserves a limited number of network states in multiple neural units but is not equivalent to the original RNN, which can be trained for an infinite number of iterations. Figure 3.2b is equivalent to three iterations with the RNN depicted in figure 3.2a. Each node in the iterative net is represented respectively by four blue, white and green nodes in the four layers of the layered net. One time-step marks parallel state transitions of neural units in the same layer, whereas the states of different layers are set

sequentially starting from the input layer. The state of the blue node in the second layer from the bottom depends on the states of the green and white nodes in the first layer. The state of the blue node in the third layer depends on the states of the green and white node in the second layer, which in turn depend on the states of the blue and white node and the blue and green node in the preceding layer.



(a) Synchronous Iterative Net

(b) Layered Net Equivalent to 3 Iterations

Figure 3.2: Mapping a Recurrent Neural Network to a Layered Net. *Three time-steps in a fully recurrent network with three neural units are mapped to a layered network. The corresponding weights are shared over all layers.*

3.4 Learning Representations via Backpropagation and Stochastic Gradient Descent

The inability of single-layer perceptrons [46], an early representative of artificial neurons, to model non-linearly separable functions was shown by Minsky and Papert [34] but resolved by multi-layer perceptrons and training through backpropagation and gradient descent. Subsequent paragraphs describe the learning procedure as introduced by Rumelhart et al. [48].

For an artificial neural network to serve as a learning model for distributed word representations, a procedure is required where language regularities are extracted from a large training set of text. Neural network learning is covered by various literature [48, 49] for the case of layered networks and an arbitrary task domain. Regularities come to be encoded - after a sufficient amount of training iterations - in the multiple feature dimensions of the internal weight matrices which can then be used for further analysis of similarities and differences. Training examples can be created directly from text segments and consist of input and output pairs, i.e. a word and its context in arbitrary order (for the corresponding architectures in word2vec, see also section 3.5). The objective is to achieve consistency between the output produced by a forward pass through the net and the desired output, defined by the training example.

As known from section 3.2, signals from an arbitrary number of active preceding cells i

reach a cell j with each a certain weight w_{ji} and are first accumulated to the total input x_j:

$$x_j = \sum_i y_i w_{ji} \tag{3.1}$$

The output of cell j, denoted y_j, is computed by a non-linear function with bounded derivative, in this case the *logistic function*[1] (sometimes also referred to as *sigmoid function*):

$$y_j = \frac{1}{1 + e^{-x_j}} \tag{3.2}$$

A finite set of $(input, output)$ pairs enables the definition of the total squared error E between computed and target output:

$$E = \frac{1}{2} \sum_p \sum_j (y_{j,p} - d_{j,p})^2 \tag{3.3}$$

Variables p and j each define an index over training pairs and output units, respectively. The desired or target output $d_{j,p}$ is subtracted from the computed output $y_{j,p}$ for the specific training pair and unit, the result squared.

Reducing the error equals learning the correct representations from a given training set, which can be approached by *stochastic gradient descent*. To calculate the value by which to update the weights it is necessary to compute the partial derivatives with respect to each weight, also referred to as the backward pass from the top layer of the neural net back to the bottom layer. Computation of the partial derivative with respect to each weight, required for gradient descent optimisation, equals computing the sum of partial derivatives for each training pair. A single training pair can each be computed starting from the output layer as follows:

$$\frac{\delta E}{\delta y_j} = y_j - d_j \tag{3.4}$$

y_j denotes the state of the output unit for a particular training pair, computed in the forward pass, and d_j its desired state as defined by the training pair. The chain rule can be applied, yielding, after differentiation and substitution, the derivative with respect to the input state of unit j (x_j):

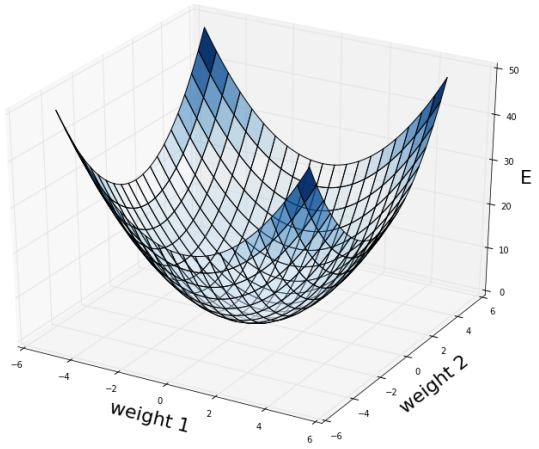
$$\frac{\delta E}{\delta x_j} = \frac{\delta E}{\delta y_j} \cdot \frac{dy_j}{dx_j} = \frac{\delta E}{\delta y_j} \cdot y_j(1 - y_j) \tag{3.5}$$

The equation expresses the effect of the total input x_j on the error which can be formulated with respect to a specific weight w_{ji} by means of equation 3.1:

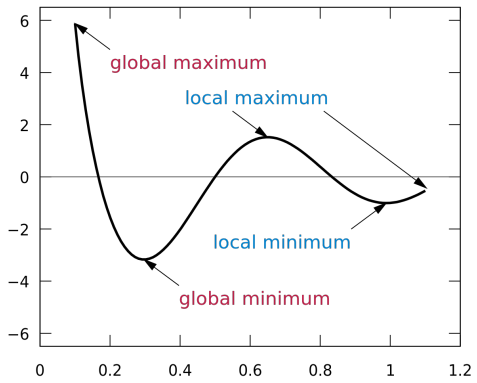
$$\frac{\delta E}{\delta w_{ji}} = \frac{\delta E}{\delta x_j} \cdot \frac{\delta x_j}{\delta w_{ji}} = \frac{\delta E}{\delta x_j} \cdot y_i \tag{3.6}$$

There are different options of how to use $\delta E / \delta w$. The weights can be updated after each training pair, after all training pairs or after single batches of the training set. More updates

[1] A step-wise derivation and more detailed characteristics of equation 3.2 can be found in the appendix on page 87



(a) Error Surface of a Linear Neuron with Two Input Weights



(b) Multiple Local Optima for the Case of Non-Linear Neurons. *Reprinted from Wikimedia Commons, by KSmrq at English Wikipedia, CC-BY-SA-3.0*

Figure 3.3: Error Optimisation with Gradient Descent. *For linear problem spaces, there is only one minimum. If the problem is non-linear as in the case of multiple neural layers used for learning representations, the algorithm can get caught in local minima. To compensate for this effect, the learning procedure can be applied to multiple random initialisations of the network.*

signifies more computations, as we will see for the comparison of model architectures in section 3.5, while accumulating weights requires additional memory. Described in [48] is the accumulation of errors over all training pairs, whereas later approaches use the first or the third option mainly for the reason they integrate well with parallel training.

The remaining decision is that by which amount to update the weights, i.e. Δw. A simple but effective improvement over subtracting a constant fraction of the respective error $\delta E/\delta w$ is that of applying a *learning rate* α in a convex combination of the error gradient and the previous update:

$$\Delta w(t) = -\epsilon \frac{\delta E}{\delta w(t)} + \alpha \, \Delta w(t-1) \tag{3.7}$$

Equation 3.7 describes how a weight w is updated at time step t (in the case of [48], once after every complete iteration over training pairs). ϵ denotes a constant factor between 0 and 1, the learning rate α, also between 0 and 1, determines the relative contribution of the previous gradient to the weight change.

One side-effect of applying stochastic gradient descent to learn internal representations was already mentioned in the context of multi-layer perceptrons: Dealing with a non-linear problem, gradient descent optimisation can get caught in local minima. Figure 3.3 illustrates the difference between the two types of error surfaces. While in the linear case (fig. 3.3a), there is only one minimum, in the non-linear case there can be multiple minima. Gradient descent approximates to a minimum depending on the starting point. If, for the simplified assumption of a two-dimensional space, the starting point in the described learning procedure lies somewhere between the two local maxima in figure 3.3b, the algorithm may find only the

local minimum.

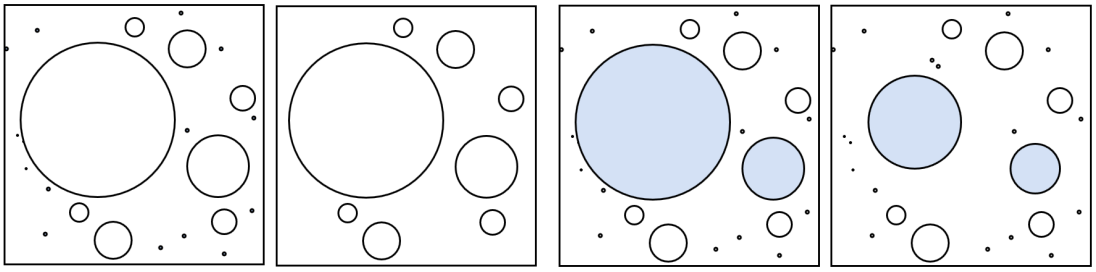
3.5 Word2Vec

Word embedding techniques define all approaches to construct multi-dimensional vector representations. Previously, we discussed the parallels between linguistic and neural structure, indicating that machine learning with artificial neural networks might be the adequate means to model natural language. As a reference and basis for our experiments, we concentrate on the detailed analysis of the mechanisms in the toolkit *word2vec*, which applies adaptive weight learning to a simplified neural network.

Word2Vec, introduced by Mikolov et al. [31] and publicly available since 2013, comprises a class of simple neural network models that produces semantic feature vector representations for each token in an unlabelled training set. Besides two general architectures and different learning objectives, it offers several parameters which enable variations in training as well as choices between optimisation techniques such as hierarchical softmax and negative sampling. The following provides an analysis of the components contributing to efficient training. The selection of these so-called hyperparameters has been identified as the main reason for the toolkit's success. Similar flexibility added to models such as Singular Value Decomposition (SVD) or Positive Pointwise Mutual Information (PPMI) has shown to achieve similarly large improvements [28] but will not be considered in favour of word2vec's computational efficiency.

Construction of Training Pairs

According to section 3.4, regardless of the chosen network architecture, training pairs must be constructed from text. The following is filtered from the text before constructing any training pairs (see fig. 3.4): words with frequency lower than a minimum threshold, referred to as *subsampling*, and a subset of occurrences of words with especially high frequency, referred to as *downsampling*.



(a) Subsampling of Words with Minimal Occurrence

(b) Downsampling of Frequent Words

Figure 3.4: Sampling of Training Data. *Words are represented by circles with a radius proportional to their frequencies. Effects of both methods are visualised separately for clarity. Applied consecutively as in popular implementations, subsampling has indirect influence on downsampling by a reduced total word count, i.e. setting a lower threshold and potentially downsampling more words.*

The intuitive assumption is that words with minimal occurrence are likely to be the product of misspellings or incorrect text processing. In any case, no meaningful representation would emerge from the small number of occurrences - they can thus be ignored. Downsampling is based on the assumption that tokens which occur very often are most probably words like *and* or *the* and exhibit only low information value. Randomly removing occurrences of the same word with a probability of

$$P(w_i) = 1 - \sqrt{\frac{t \cdot \sum_w count(w)}{count(w_i)}} \tag{3.8}$$

for word w_i and a threshold t in the order of 10^{-5} results in an overall more even distribution while preserving the frequency ranking among vocabulary items. Among other downsampling possibilities, equation 3.8 is the result of a heuristic analysis reported in [32]. On the one hand, it ensures increased information density in subsequently formed contexts. On the other hand, the representation of the respective word remains more or less intact as a considerable amount of remaining occurrences guarantees that the neural net does learn a suitable representation. To learn the meaning of words in compliance with the distributional hypothesis (see also page 9), training pairs are formed from words and their surrounding context. Whereas 'word' always applies to the middle word in the window under observation, the size of the latter can vary. The maximum distance to the focus word, denoted as r, defines a range from 1 to r in which contexts are uniformly sampled. Instead of applying a weighting scheme to a context window as is the case in other approaches such as GloVe[2] (short for "Global Vectors"), the largest context word distance c for each training pair is randomly chosen from a range between 1 and r, leading to a context of size $2 \cdot c$. This allows for more efficient computation and, applied to a huge training set, corresponds roughly to weighing context words by $\frac{d}{r}$, their distance to the focus word d divided by the maximum distance.

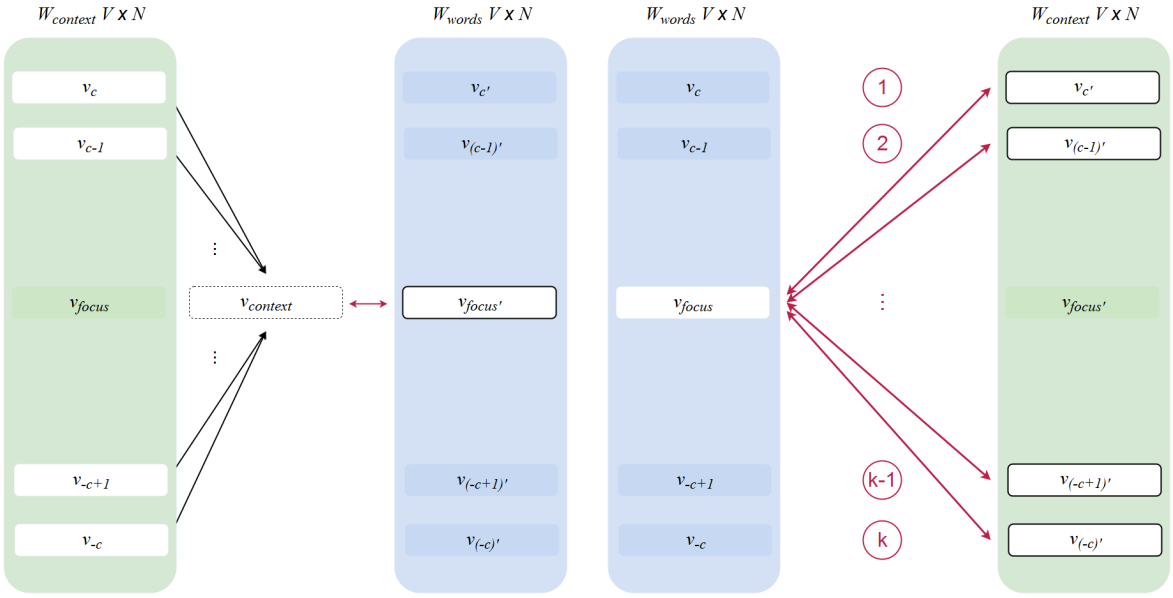
3.5.1 Neural Network Architectures and Update Frequency

Mikolov et al. [32] introduce two general network architectures *Continuous Bag-of-Words* and *Skip-gram* for word2vec with reverse application of $(word, context)$ training pairs to input and output. An essential difference poses the gradient descent update frequency, which is considerably higher for Skip-gram. Below, the two network architectures are discussed in detail.

The general architecture of *Continuous Bag-of-Words* (CBOW) is a neural network model with shared projection layer which approaches the task of learning semantic word vectors from the perspective of predicting a word given its context. The model is depicted in figure 3.5a, where the words of a context of size $2c$, denoted w_{i-c} to w_{i+c} without the middle word w_i, form the active input and w_i is taken as the desired active output of the training instance. Parameter c is defined as the maximum distance to w_i.

On initial training, an $V \times N$ matrix $W_{context}$ is randomly initialised with each row representing a vocabulary term in V and each column representing an unlabeled feature dimension.

[2] embedding technique based on explicit global matrix factorisation, introduced by Pennington et al. [42]



(a) Continuous Bag-of-Words (CBOW) (b) Skip-Gram

Figure 3.5: Word2Vec Neural Network Model Architectures

The reduced number of feature dimensions, i.e. the cardinality of N can be freely selected as a parameter.

In remembrance of section 3.2, the first step forward is an aggregation of the inputs. For the default CBOW behaviour, this corresponds to the average of weighted inputs, i.e. the row vectors in $W_{context}$ for the context words $\{w_{-c}, ..., w_c\}$:

$$v_{context} = \frac{1}{2 \cdot c} W \cdot \sum_{i=1}^{V} w_i \cdot v_i = \frac{1}{2 \cdot c} W \cdot \sum_{i=1}^{2 \cdot c} v_i \tag{3.9}$$

In equation 3.9, W denotes the input weight matrix and c the maximum distance to the focus word. A second option given in word2vec is a simple summation of the inputs.

The weights are updated once in a single gradient descent step for the complete training pair.

To summarise the previous steps and relate CBOW initialisation with the training pair construction described above, we consider the following (erroneous) sentence as an example:

$$Gary\, the\, cat\, spots\, a\, mouse\, betweeen\, the\, flowers\, and\, chases\, it\, through\, the\, garden.$$

In a real corpus, there are various points in the text where punctuation such as periods are used for other purposes[3] than sentence boundaries. A sentence tokeniser thus has to handle these with language-specific attention. After sentence tokenisation, we obtain 16 tokens:

$$'Gary','the','cat','spots','a','mouse','betweeen','the','flowers','and','chases','it',$$
$$'through','the','garden','.'$$

[3] we omit these here for clarity

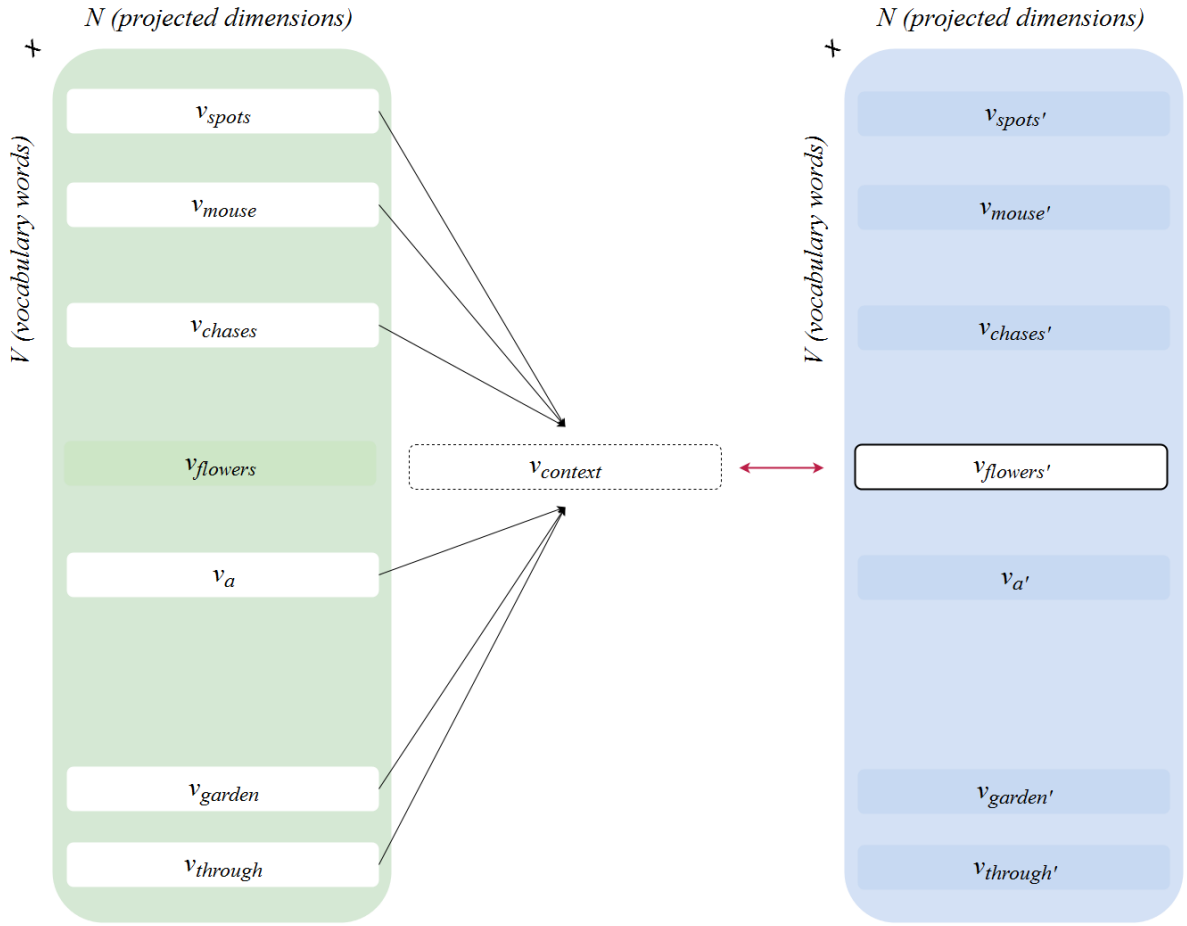


Figure 3.6: Training of $(flowers, \{spots, a, mouse, chases, through, garden\})$ with CBOW Model. *The gradient descent of errors is conducted once, for one context vector, summarised or averaged from all words in the context window.*

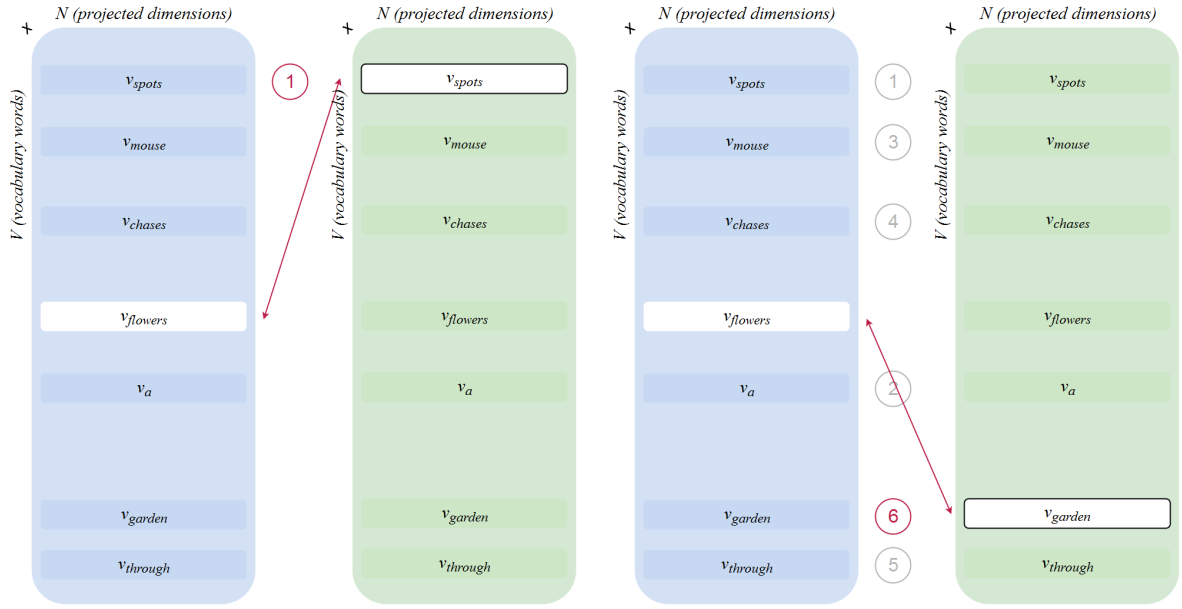
We filter for words only, which removes punctuation and quotes, leaving us with 15 tokens:

$$'Gary', 'the', 'cat', 'spots', 'a', 'mouse', 'betweeen', 'the', 'flowers', 'and', 'chases', 'it', 'through', 'the', 'garden'$$

Now we can apply subsampling and downsampling as described above. By subsampling, we remove $betweeen$, the word with incorrect spelling. In a large corpus, this would help to keep the vocabulary at a realistic size. The representation of the word $between$ would be learned from its other occurrences. Downsampling randomly removes occurrences with a probability proportional to the word count in the corpus. Accordingly, overall occurrence of the words the, and, a and it can be assumed very high in an English corpus. A possible result is the following subset of 9 tokens:

$$'Gary', 'the', 'cat', 'spots', 'a', 'mouse', 'flowers', 'chases', 'through', 'garden'$$

Considering a setting with dynamic context window of size 5, i.e. a maximum distance of 5 to the focus word, we construct a sample training pair for the focus word $flowers$. As



(a) First training step

(b) Last training step

Figure 3.7: Training of $(flowers, \{spots, a, mouse, chases, through, garden\})$ with Skip-Gram Model. *The gradient descent of errors is conducted separately, i.e. 6 times for each context word.*

a possible random choice of the window size for this particular training pair, we assume 3 (normal distribution between 1 and 5), yielding a tuple with $flowers$ and its context of 6 words:

$$('flowers', \{'spots', 'a', 'mouse', 'chases', 'through', 'garden'\})$$

The example training pair contains 7 distinct words. An overall corpus would have a considerably larger vocabulary. For the given training example, we set up CBOW with the randomly initialised context word vectors as the input and the focus word vector as the desired output. The described setup is depicted in figure 3.6.

For the example in figure 3.6 this results in the words $the, mouse$ being weighted by $\frac{1}{2}$ and cat, the (in this case the second occurrence of the), which are closest to the focus word $chases$ by $\frac{2}{2}$, i.e. 1. It is important to note any steps to reduce the corpus size are taken prior to forming the context windows.

The *Skip-Gram* model depicted in figure 3.5b was introduced in [31] as the counterpart to CBOW with a shared projection layer and inputs and outputs reversed. Here, the objective is to predict the context given a word. To represent w_i at an arbitrary position k in V, the input of all units except the k-th is set to 0, the input to unit k is 1 (active). In simplified terms, with w_i as input of a training pair, the input to the neural net is a *one-hot-encoded* vector of size V or *one-of-V-encoded* at respective position k. A $V \times N$ matrix W_{word} is randomly initialised and encodes the weights for all inputs to the projection layer. A second matrix $W_{context}$ with V rows and N columns, encodes the weights for all inputs to the output layer. The output layer consists of $|V|$ output units, with the $2c$ output units $\{w_{-c'}, ..., w_{c'}\}$ activated consecutively for the given training pair. By backpropagation of errors, the model

gradually learns an internal representation of words and their context (the rows in W_{word}), such that the total error over all training pairs between the computed output and the desired output is approximately minimised. This procedure is repeated for the whole corpus and a user-selectable number of full corpus iterations. The adaptive learning rate α further influences how much the weight is adjusted per each gradient descent update.

To illustrate CBOW, we introduced an example sentence above:

$$Gary\,the\,cat\,spots\,a\,mouse\,betweeen\,the\,flowers\,and\,chases\,it\,through\,the\,garden.$$

One possible construction of the training pair for the focus word $flowers$ was the following:

$$('flowers', \{'spots','a','mouse','chases','through','garden'\})$$

Figure 3.7 shows the respective setup with skip-gram. In contrast to CBOW, the given training pair with a context of 6 words results in 6 gradient descent updates, each minimising the error in respect to a single context word. Together with the adaptive learning rate, the resulting models differ such that a categorical decision for one over the other is hardly possible.

3.5.2 Hierarchical Softmax

The last section covered the two different model architectures available in word2vec. Additionally, there are two different objectives for which to compute the error. One possibility to model the conditional probability of an output given the input computation of the forward pass is the softmax function:

$$p(c|w;\theta) = \frac{e^{w_k \cdot w_i}}{\sum_{w_{k'} \in C} e^{w_{k'} \cdot w_i}} \tag{3.10}$$

The softmax function ensures a probability distribution such that all values assigned to single output units are between 0 and 1 and the sum over all outputs equals 1. It is possible to increase training efficiency by applying an approximation of the full softmax, namely hierarchical softmax. The method, whose application to natural language modelling was first proposed by Morin and Bengio [39], reduces each evaluation of output nodes to logarithmic complexity by representing the output layer as a binary tree. An example of a binary Huffman tree as it is constructed by the algorithm is depicted in figure 3.8. Shortest codes are assigned to most frequent words, resulting in a considerable training speedup.

3.5.3 Negative Sampling

The alternative probability computation for the output $y_k(t)$ is to generate, additionally to the set of training pairs, a set of negative samples. Negative sampling is a simplification of *noise contrastive estimation* (NCE), a general parameter estimation technique proposed by Mnih and Teh and Mnih and Kavukcuoglu [36, 37] for training neural network language models. With negative sampling, language model estimation can be reduced to a binary classification problem: samples from an empirical test set and samples of words and contexts randomly combined. A possible neurolinguistic explanation is the omnipresent concept of negation and

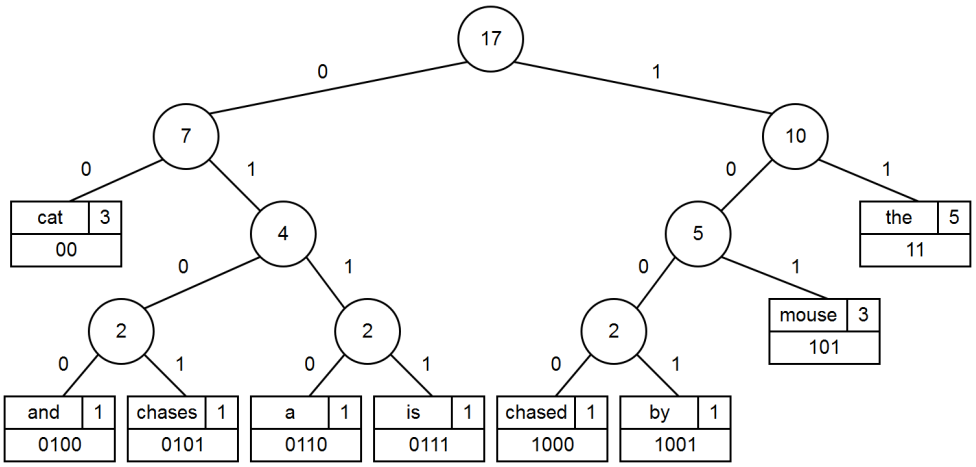


Figure 3.8: Example Word Encoding with Binary Huffman Tree.

that, learning a language, observed can be not only what occurs together, but also what is absent in the context of one another.

Following the lines of Goldberg and Levy [12], let D denote the set of $(word, context)$ pairs observed in the training corpus. The set of negative samples D' is defined by a number of k randomly generated words $word'_1, ..., word'_k$ that replace the focus word, resulting in k additional pairs per element in D:

$$D' = (word'_i, context)|(word, context) \in D, word'_i \neq word, word'_i \in V, i \in 1, ..., k \quad (3.11)$$

The probability that a pair (w, c) originates from the corpus can be formalised as $p(D = 1|w, c)$, the reverse probability respectively as $p(D = 0|w, c) = 1 - p(D = 1|w, c)$. The probability of seeing c in the context of w should naturally be close to 1, while the expected probability for c in the context of w'_i is 0 or close to 0. The resulting optimisation objective is as follows:

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(D = 1|c, w; \theta) \prod_{(w,c)\in D'} (1 - p(D = 1|c, w; \theta)) \quad (3.12)$$

Taking the logarithmic probabilities simplifies the equation to a summation function:

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log p(D = 1|c, w; \theta) + \sum_{(w,c)\in D'} \log(1 - p(D = 1|c, w; \theta)) \quad (3.13)$$

Using the definition of $p(D = 1|c, w; \theta)$ as softmax (sigmoid normalised over output variables) from equation 3.10 leads to the following equation:

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log \sigma(v_c \cdot v_w) + \sum_{(w,c)\in D'} \log \sigma(-v_c \cdot v_w) \quad (3.14)$$

3.5.4 Parallelisation

In large data processing, parallelisation plays an important role. Word2Vec supports parallelisation in the form of *asynchronous stochastic gradient descent* combined with an adaptive

learning rate procedure called *AdaGrad* [31]. This enables distribution of training across multiple instances and with it increases the efficiency of unsupervised feature learning even with a large number of parameters. Details and evaluation are provided by Duchi et al. [7] regarding AdaGrad and Dean et al. [6] for the combination of techniques as optimisation method in the framework DistBelief. The weight matrix, i.e. the inner representations learned during training are shared over all instances, fetched and updated by the gradients in an asynchronous manner. This leads to occasional overwrites when multiple model instances access the same row of weights simultaneously. In practice, though, this coincidence occurs only rarely and apparently has no negative effects while offering substantial speed-up in contrast to a synchronous implementation with locks [6]. Parallelisation can be customised through the choice of the initial learning rate, the minimum learning rate and the number of threads for the distribution of training examples.

3.5.5 Exploration of Linguistic Regularities

As suitably formulated by Levy and Goldberg, while training neural word embeddings does not discover novel patterns, it is "doing a remarkable job at preserving the patterns inherent in the word-context co-occurrence matrix" [26, p.172]. Individual tuning allows to emphasise and work out regularities at multiple layers.

To analyse patterns of e.g. the relation of words in their masculine form towards their feminine form or a country and its capital (distributed over all projected dimensions), it is possible to look at the commonalities of different word pairs representing that relation. An example of a projection to two-dimensional space via principal component analysis is depicted in figure 3.9. The vector offsets describe the relation from countries to their capital as encoded in 1000-dimensional skip-gram vectors.

With a good approximation over a significant set of pairs, it should thus be possible to retrieve good results on terms sharing this relation. The machine **has learnt** to distinguish between genders. Due to the fact that the human brain is a neural network of an incomparably higher order, it is obvious that one simplified model cannot perform equally well on all tasks. As a consequence it is essential to identify the similarity tasks important in the respective context. The methods detailed in the following suggest an additional way to evaluate what is otherwise hidden in dimensionality and size of the model.

In this context it is worth having a closer look at how one of four vectors building an analogy is estimated. The method first introduced by Mikolov et al. [31] and later referenced as *3CosAdd* by Levy and Goldberg [26] approaches this from the perspective of maximising the similarity between the word to be predicted and the linear combination of the analogous word pair and the single candidate. The formal definition, denoted as equation 3.15 is as follows:

$$\arg\max_{b^* \in V}(cos(b^*, b) - cos(b^*, a) + cos(b^*, a^*)) \tag{3.15}$$

The objective is to predict the word b^* such that "a is to a^* as b is to b^*". In [26], Levy and Goldberg introduce a multiplicative combination of cosine similarities by the name *3CosMul*:

$$\arg\max_{b^* \in V} \frac{cos(b^*, b) \cdot cos(b^*, a)}{cos(b^*, a) + \epsilon} \tag{3.16}$$

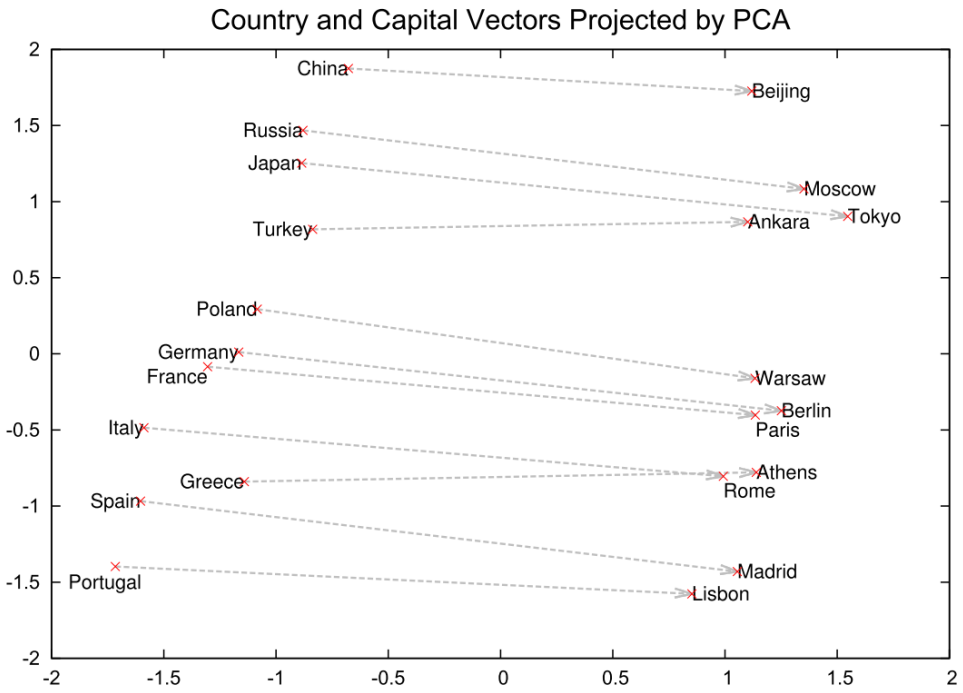


Figure 3.9: Analogous Country - Capital Regularity. *Projected via two-dimensional PCA. Reprinted from Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality" [32, p.4]*

ϵ is a constant set to 0.001 to prevent a division by zero. Results reported since the introduction of the refined method *3CosMul* consistently confirm improved results and the method is part of the common reimplementation *gensim* (Python).

4 Clustering Techniques

In the previous chapter we reviewed the historical and technical details of how the interoperation of biological neurons is mimicked in order to learn substantial regularities in highly complex systems, in this case the linguistic world of concepts as represented in a concrete training corpus. Significant features of this approach are computational efficiency, robustness against errors and the exceptional capabilities of parallel reduction in comparison to statistical models. For extended query-specific support, it requires the further step of extracting the structure in a subset of word vectors, i.e. partitioning into groups according to semantic feature similarity. This local structure on word level can be utilised to provide efficient navigation to a minimal set of relevant query results. Before evaluating word embeddings with different parameter configurations, we review options of semantic grouping. Section 4.1 provides a general categorisation of clustering approaches. Section 4.2 illustrates the challenges and that arise when dealing with high-dimensional data and possible ways to address them.

4.1 Categorisation

Although all clustering approaches perform an unsupervised partitioning, i.e. propose an organisational scheme for a set of data, there is no precise definition of a cluster as such. However, knowledge of the different types and accompanying strengths and weaknesses is essential for choosing the algorithm best suited for the given use case. To facilitate a basic understanding, the following constitutes a common distinction of group-building algorithms, including illustrations, in accordance with standard literature such as [15, 29] and [3].

One substantial characteristic is the type of structures a clustering algorithm produces, roughly divisible into *flat* and *hierarchical* clusters. As a preliminary step to flat clustering, a first partitioning of the data set is performed. In general, this partitioning is not required to consider the distribution of data objects - more typically, it follows a randomised approach in order to provide a fair starting point for the algorithm. Based on the initial clusters, flat clustering determines the best partitioning into a number of unrelated groups through iterative reassignment. The algorithm stops when a certain stopping criterion is met. Flat clustering algorithms, however, can also be applied as a secondary step to hierarchical clustering. In this case, the aim is to further improve the first grouping. Iterative reassignment is directly executed on clusters produced in the previous step. One popular representative of flat clustering is *K-means*, which assigns n observations to a predefined number of k clusters according to their closest centres. If a partitioning is not provided in advance, k vectors $f_1, ..., f_k$ are randomly distributed over the problem space, serving as centres for the initial assignment of observations.

Algorithm 4.1 K-Means. *Adapted from Manning and Schütze "Foundations of Statistical Natural Language Processing" [29, p. 516]*

Require: a set of observations $\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$
Require: a distance measure $d : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$
Require: a function for computing the mean $\mu : \mathcal{P}(\mathbb{R}) \to \mathbb{R}^m$
  Select k initial centres $f_1, \ldots, f_k$
  while stopping criterion is not true **do**
    for all clusters c_j **do**
      $c_j = \{x_i | \forall f_l d(x_j, f_j) \leq d(x_i, f_l)\}$
    end for
    for all means f_j **do**
      $f_j = \mu(c_j)$
    end for
  end while

In case of an already performed hierarchical clustering and, generally, after the first initial step, the center is recomputed as the *centroid* or mean μ of its members. Algorithm 4.1 describes the general algorithmic procedure.

Hierarchical clusters can be built either top-down (*divisive*), starting with all observations in one cluster, or bottom-up (*agglomerative*), starting with each data object as a separate cluster. Accordingly, a measure needs to be specified by which to split the existing groups or merge the members of two groups into one. The algorithm principally terminates when all groups consist of a single observation or the resulting group contains the complete set of observations, respectively. The emerging organisation after complete (de)composition is that of a hierarchical tree which can be cut at arbitrary level, the unconnected branches posing the individual clusters, the nodes the objects belonging to the respective cluster. Alternatively, a stopping criterion can be defined for a certain number of clusters or a certain distance between the clusters, but in contrast to flat clustering, it is not obligatory to specify the number of clusters in advance. The general algorithmic procedures for an agglomerative hierarchical clustering is described in algorithm 4.2.

So far we have assumed the distance function as given. However, the choice of distance measure constitutes a crucial factor for the resulting clustering. A common choice for continuous variables is defined by the L_r-norms, $r \geq 1$, given in equation 4.1:

$$d_{ij} = ||x_i - x_j||_r = \left(\sum_{k=1}^{p} |x_{ik} - x_{jk}|^r \right)^{\frac{1}{r}} \tag{4.1}$$

As the measure of choice for distributed word representations, which are normalised with the squared L_2-norm or squared Euclidean distance, the equation can be written as follows:

$$d_{ij}^2 = ||x_i - x_j||_2 = \sum_{k=1}^{p} |x_{ik} - x_{jk}|^2 \tag{4.2}$$

With this metric we can estimate the distances between single observations, but for clusters with more than one member we need to further define the function by which to compute

Algorithm 4.2 Agglomerative Hierarchical Clustering

Require: a set of observations $\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$
Require: a distance function $d : \mathcal{Y} \times \mathcal{Z} \to \mathbb{R} \quad | \quad \mathcal{Y}, \mathcal{Z} \subset \mathcal{X}$
  for all observations x_j **do**
    $c_j := \{x_j\}$
  end for
  $C := \{c_1, \ldots, c_n\}$
  $j := n + 1$
  while $|C| > 1$ **do**
    $c_j := c_{n1} \cup c_{n2} \quad | \quad d(c_{n1}, c_{n2})$ minimal in C
    $C := C \setminus \{c_{n1}, c_{n2}\}$
    $C := C \cup c_j$
    $j = j + 1$
  end while

the distances between the clusters. For the partitioning algorithm k-means discussed above, we already mentioned the computation of the centroid as center of each cluster, used as the point of comparison. This is in fact one possible computation of a group distance. A simple alternative often applied to hierarchical techniques is *single linkage*: instead of computing a center, the closest member of a cluster, i.e. the *nearest neighbour* towards the second participant in the comparison, is considered. A known disadvantage of this definition is that very large clusters can be formed where data would better be represented by two or more smaller clusters in close proximity. *Noise* or *outliers* can blur the borders and act as single stepping stones.To deal with this problem, other techniques have been developed, e.g. *complete linkage*, which in turn considers the largest individual distances, resulting in rather small clusters. As a compromise, *complete linkage* computes the average distance.

We follow the notation in [15] and define a distance function d in dependence of two clusters to be merged (P and Q) and another cluster R towards which to compute the distance. Function d is given in equation 4.3. Listed by name in table 4.1 are possible definitions of δ_1 to δ_4.

$$d(R, P + Q) = \delta_1 d(R, P) + \delta_2 d(R, Q) + \delta_3 (P, Q) + \delta_4 |d(R, P) - d(R, Q)| \tag{4.3}$$

A more cautious approach which in practice frequently yields improved results is the *Ward* algorithm. In contrast to linkage computations, Ward does not simply merge (or in the top-down approach, split) groups according to smallest (largest) distance. Instead, the condition is not to increase the variation inside a group too much, i.e. to keep the heterogeneity at an acceptable level. With the function as defined by equation 4.3 and the respective entry in table 4.1, Ward is related to the centroid (geometric distance). Both are more sensitive to outliers than the first examples, whose drawbacks were outlined above.

Name	δ_1	δ_2	δ_3	δ_4
Single linkage	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete linkage	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
Average linkage (unweighted)	$\frac{1}{2}$	$\frac{1}{2}$	0	0
Average linkage (weighted)	$\frac{n_P}{n_P+n_Q}$	$\frac{n_Q}{n_P+n_Q}$	0	0
Centroid	$\frac{n_P}{n_P+n_Q}$	$\frac{n_Q}{n_P+n_Q}$	$-\frac{n_P n_Q}{(n_P+n_Q)^2}$	0
Median	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0
Ward	$\frac{n_R+n_P}{n_R+n_P+n_Q}$	$\frac{n_R+n_Q}{n_R+n_P+n_Q}$	$-\frac{n_R}{n_R+n_P+n_Q}$	0

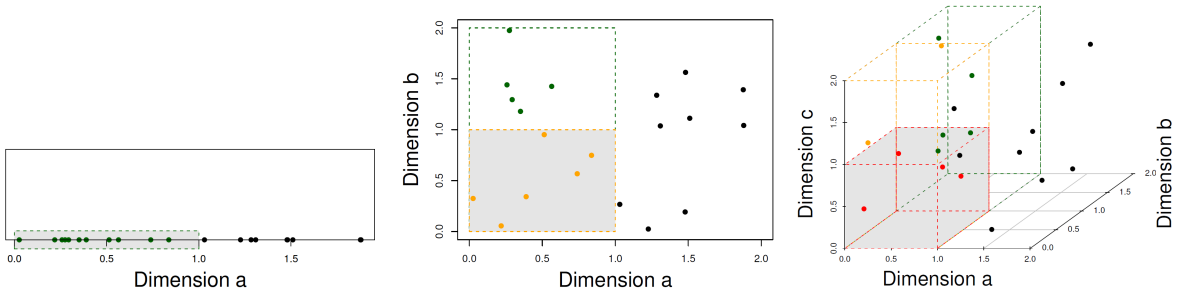
Table 4.1: Computations of Group Distances. *Adapted from "Applied Multivariate Statistical Analysis" by Härdle and Simar [15, p. 393]*

4.2 The Curse of Dimensionality

Challenges accompanying high dimensionality are well known in many fields of scientific research and commonly referred to as the *curse of dimensionality*. Specifically, it designates the problems and manifestations when approaching the organisation of more than three-dimensional settings, increasing in severity with each additional dimension. Figure 4.1 offers a good illustration of how data sparsity increases with each additional dimension. Considering one unit in a projection of objects to one-dimensional space, contained are 11 objects. If we extend our observation to two-dimensional space, we can see that only 6 objects lie in the same unit, whereas for substantial differences on the dimension b, 5 objects fall into a separate unit. This increases exponentially as we add a third dimension to the observation, leading to only 4 objects in a one unit bin.

From the current state of knowledge, a good semantic representation of words by their context can be achieved with projections to a low-dimensional space (compared to vocabulary and semantic facets), but still ranging from 300 to 500. This was already indicated by a performance curve for a closed question similarity test set by Landauer and Dumains [24] in relation to latent semantic analysis, but is confirmed in recent publications on neural word embedding models [28, 31]. Although the vector space of feature dimensions is less *sparse* than for approaches without projection such as it takes place during neural network training, even the (relatively) dense space of word context features is heavily affected.

As a side note we want to point out that the mentioned claims of a good semantic representation are mostly based on the **evaluation** of how well word similarities and the analogous relation between pairs are preserved in the projected feature dimensions of distributed language models. A closer look at evaluation methods, imposing yet another set of concerns, is going



(a) 11 objects in one unit bin (b) 6 objects in one unit bin (c) 4 objects in one unit bin

Figure 4.1: The Curse of Dimensionality. *Data sparsity increases with each additional dimension. Similarity of one object to another does not provide insight into how these are similar, two objects similar to another can be totally different from each other. These challenges for clustering are common to all application areas with high-dimensional data. Reprinted from "Subspace Clustering for High Dimensional Data: A Review" by Parsons et al. [40]*

to follow in section 5.3. The actual measure of comparison between word vectors - and no explicit substitutes are discussed in existing literature on distributed word representations - is cosine similarity and its additive and multiplicative combination as detailed previously on page 23.

What are the main challenges in finding clusters with respect to distributed word representations? We have briefly outlined for the general case that with multiple dimensions, measures such as cosine similarity become meaningless. By cosine similarity, we can measure the mathematical proximity between two points, but we do not gain any insight into how word *x* is closely related to word *y*. Suppose we have the two words *Katze* (cat) and *Hund* (dog) we expect to find similarity encoded as to the fact that both are animals and common pets. A common saying is that cats live a more independent life whereas dogs can be the most loyal four-legged friends. In terms of vector similarity, assuming the underlying training corpus represents general knowledge, we expect a high overall similarity due to the shared formerly mentioned features and bigger differences in certain subspaces related to the independent life of cats as opposed to the one of dogs in which nature it is to live in packs.

With innumerable facets of meaning encoded within a few hundred dimensions, it is easy to imagine that the characteristics *loyal* and *independent*, besides their being not universally agreed on, have to share dimensions with other features. More complex, we discussed the derivation of the notion *distributed* from the representation of meaning as a pattern. As such, *loyal*, *independent* or *animal* cannot be deduced from one dimension but relate to a general shift of many dimensions. The distribution of meaning over many dimensions, i.e. the meaning by which we want to group a local subset, imposes a general problem on conventional feature algorithms: As soon as we normalise vectors to euclidean space, we loose important information (compare fig. 4.2). We cannot expect clusters to lie all parallel to these axes, thus some of the clusters get lost. An example where this occurs is illustrated in figure 4.2. By either considering dimension x or dimension y, a clustering algorithm will miss two clusters, namely c_{ab} and c_{ad}, instead detecting both as part of c_a. As displayed, however, both form two

clusters, clear to see with a good eye. With high-dimensional word representations, it is hardly possible to imagine a use case where subsets or combinations of features are not relevant for different clusters.

Common techniques to better deal with an initial high dimensionality are *feature transformation* and *feature selection*. Feature transformation attempts to combine correlating attributes into one, whereas feature selection identifies most relevant features from the task set and discards them. As we want to apply clustering to a local subset surrounding the query term, we consider both methods for dimensionality reduction a potentially useful extension.

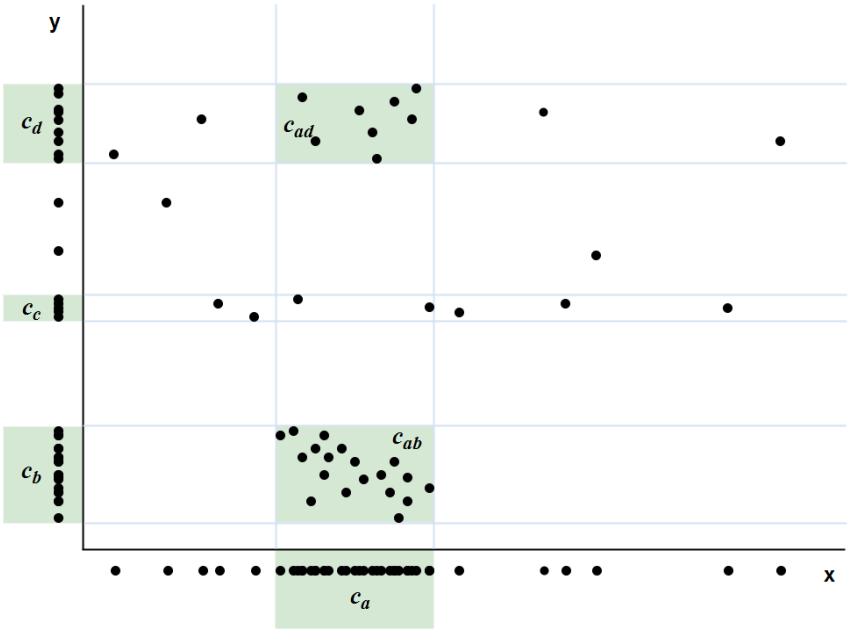


Figure 4.2: Clustering in Multi-Dimensional Space. *Applying normalisation such as the L_n-norms can lead to problems with clusters in subspaces, here c_{ab} and c_{ad}.*

5 Training and Evaluation of Neural Embedding Models

In the previous chapters we provided the theoretical foundations for our practical studies with *gensim*, an implementation of the toolkit *word2vec* in the programming language *Python*. An overview of milestones in the historical developments of distributed language models and machine learning with simplified artificial neural networks was given in chapter 3. In section 3.5, configuration parameters for model training with word2vec were analysed in detail. Thereafter, we discussed the problems and capabilities of clustering analysis with respect to applicability to the resulting multi-dimensional word representations in chapter 4. In the following we approach the central objective to identify improvements and extensions for enterprise search based on distributed language models by empirical evaluation. Sections 5.1 and 5.2 document the technical setup and design considerations. In section 5.3 we describe measurement goals and a compilation of test sets. We use these for internal and external comparison to official benchmarks in section 5.4, converging in the choice of models for cluster analysis in chapter 6.

5.1 Technical Setup

For application-oriented research on distributed word representations within the company interface projects, the most important basis is a platform-independent, closed and versioned development environment. Second requisite is a software stack that allows to work interactively when investigating the effects of varying processing from the step of data collection to the final steps of evaluation and visualisation. The former is ensured through a distributable virtual image under version control, for the latter, the choice of programming language constitutes an important criterion. *Python* is known to offer good support for natural language processing and machine learning tasks, not least because of libraries such as *NLTK*[1], *SciPy*[2] [21], *matplotlib*[3] [20] and *scikit-learn*[4] [41]. High-performance open-source data analysis tools include *pandas*[5] [30] and *NumPy*[6] [57]. Regarding our practical studies, we concentrate on the use of

[1] `http://www.nltk.org/` [accessed 15 June 2016]
[2] `http://www.scipy.org/` [accessed 4 July 2016]
[3] `http://matplotlib.org/` [accessed 4 July 2016]
[4] `http://scikit-learn.org/stable/` [accessed 15 June 2016]
[5] `http://pandas.pydata.org/` [accessed 15 June 2016]
[6] `http://www.numpy.org/` [accessed 15 June 2016]

gensim[7] [44] for training our distributed language models, a Python natural language processing toolkit with an implementation of word2vec, which, like most of the mentioned libraries, makes use of more efficient processing in its computation-intensive steps with *Cython*[8]. To facilitate efficient coding and collaboration, we use *Jupyter* with *IPython*[9] [43], an interactive shell with language-specific support and additional documentation features. Code and documentation blocks can be flexibly arranged and executed within a notebook. Different *magic commands*[10] give extended control, e.g. to influence library reloading, monitor variables or execute shell commands. For evaluation of the collected evaluation test scores, we finally decided to port everything to *Excel*, made easy by the respective conversion function in pandas. The notebook environment contributed to the seamless transfer but did not prove the right choice for our needs in maintaining and interacting with the plain evaluation test scores.

5.2 Model Training

High value of the language modelling techniques combined in *word2vec* lies in its open design choices made readily accessible through various hyperparameters. But even these are just a subset of decisions to be made - additionally, a number of factors have to be considered in regard to preprocessing of the texts. The following provides an overview of selected parameter configurations and test series considered worthy of pursuit under the given objective. All models are trained using the Python library *gensim*, its concrete parameters referenced and related to their algorithmic counterparts in section 5.2.6.

5.2.1 Corpus

A large quantity of training examples, generated from text data, and the linguistic quality thereof are essential for obtaining high-quality word representations. Besides a certain minimum size, requirements for the corpus in respect to our task reside in it reflecting different use cases from corporate domains while preserving universal comprehensibility of semantic relationships.

We use articles of the German news magazine *Der Spiegel*[11] for multiple reasons. In the context of enterprise search data amounts vary greatly depending on company size and domain. However, given the increasing use of software tools and email for business correspondence, there is still an upward trend. In the case of *Der Spiegel* print magazine, articles from 1947 until 12 months ago (328 079 documents from full years 1947-2014, after clean-up) are freely accessible from an archive. An overview of the top 10 most frequent and rarest words after removal of those with an occurrence lower than 5 (see also subsampling on page 16) offers table 5.1.

On the one hand, table 5.1a confirms the strong proportional representation of words with low information value such as articles, propositions and conjunctions, which illustrates the

[7] `https://radimrehurek.com/gensim/` [accessed 15 June 2016]
[8] Python-compatible compiler offering C-like performance: `http://cython.org/` [accessed 15 June 2016]
[9] `https://ipython.org/` [accessed 3 July 2016]
[10] `http://ipython.readthedocs.io/en/stable/interactive/magics.html` [accessed 15 July 2016]
[11] available from `http://www.spiegel.de/spiegel/print/` [accessed 20 April 2016]

Token	Count
der	6555686
die	6469120
und	4339480
in	3444741
den	2636452
zu	2062135
von	1899262
das	1826126
mit	1695645
sich	1683576

(a) Top 10 Most Frequent Tokens

Token	Count
Fachbetriebe	5
Eigenvertrieb	5
Stecherei	5
Symphonie-	5
Morgenkonferenzen	5
Bienenvater	5
Verzweiflungsruf	5
pikanteste	5
davonjagt	5
89jaehrig	5

(b) Top 10 Rarest Tokens

Table 5.1: Top Most Frequent and Rarest Corpus Tokens. *Frequencies rankings refer to the corpus of unnormalised single tokens after removal of punctuation. Tokens with less than a minimum count of 5 were excluded.*

necessity of removing or downsampling these words for better performance and more concentrated information in created contexts. On the other hand, compared to other sources such as mixed web corpora, we can observe a relatively high quality already for the tokens in table 5.1b. All contents are directed at a general audience with a certain education level, researched and prepared such as to meet consistent editorial standards. We can thus rely on an overall good information base, starting from which we can argue with generally intelligible examples. While an open encyclopaedia such as the German Wikipedia does offer large amounts of valuable data, there are no guarantees as to which content is sufficiently well represented and which might be overrepresented. The online encyclopaedia's size exceeds the *Spiegel* corpus with 1.945.033[12] entries roughly fivefold in comparison to the number of articles stated above. However, included in this number are many entries consisting of only a short link list or content of up to five sentences. Working with the smaller, but considerable size of a news archive and content more concentrated on common topics such as politics, sports, economics and culture imposes less critical restrictions in terms of space and time and enables a more thorough examination of parameters in multiple test series.

We removed articles with potentially interfering influence to the task at hand such as imprints. The dataset counts 119 927 350 *tokens*, i.e. symbols separated by whitespace, tab or end of line, and a vocabulary of 3 967 124 unique words or *word types* from which words with less than three occurrences were removed during creation.

A good representation for a word can be determined starting from a certain number of occurrences. However, we retain even rare words above a low minimum count of 5, following the assumption that they often are of respectively higher semantic importance and can help differentiate the context of other words. This aligns well with other studies [31] where the min_count parameter was set to a similar value, but for evaluation only tasks with words

[12] according to `https://de.wikipedia.org/wiki/Spezial:Statistik` [accessed 31 May 2016]

occurring at a volume of e.g. 100 were considered. Köper et al. [22] apply a min_count of 50 but train on a large collection of web content which can be assumed to introduce more junk - the higher parameter value is needed to remove more of the latter.

News articles cover a variety of domains, not necessarily in-depth, but sometimes also extensively as in case of political proceedings or sport events. Just like in corporate data named entities play a very important role. While the corporate context can be expected to involve mostly local entities, public figures permit an evaluation of how well these are recognised and brought into an accurate semantic relationship.

5.2.2 Data Segmentation and Ordering

There have been suggested multiple units within which to consider the context of words when training their distributed vector representations. Besides the applied dynamic context window - specified by the maximum window size, sampling rate for frequent words and minimum count for subsampling (see section 3.5), it is advisable to segment the data into semantically independent parts.

The original idea suggested also by the variable assignment in implementations such as the C implementation (*sents*) is to train on sentence batches [31]. It is easy to imagine that important aspects of a word's semantic meaning exceed certain sentence boundaries and are restricted to others. Quickly following publications report models trained on paragraphs or documents [33]. In case of the given news articles, we choose the default behaviour to shuffle document-wise and train on sentence batches. A short comparison between models with the same configuration, trained once on sentence batches of a document shuffle and once on sentence batches of a complete sentence shuffle, confirms this decision.

With training all $(word, context)$ pairs about one topic at the very beginning and no later occurrence, the representation of the semantic relations regarding this topic will vanish when trained on a large number of subsequent pairs. This effect can be reduced by shuffling the text segments, which integrates well with parallelisation. On the other hand, shuffling text segments means disregarding evolution over time. Given a product which has been renamed from the former to a newer version, the timely connection cannot be reconstructed. For the case of terms replaced over time, e.g. different version names, we consider it more important to capture the similarity and not the chronological order in which one term replaced the other. We abandon time information in favour of simplified use with the sidenote that this piece of information can indeed be reconstructed from document meta fields.

In the enterprise search context, it is the general case to have structured documents where certain parts have a certain purpose or meaning. Naturally associated therewith, some parts are less important from a general content-relating perspective. To name an example from the corpus under examination: In an article about politics, the author might be in connection with the opinion represented, but he has no direct link to the member of the party about which he writes. He certainly does not represent a strong connection between the party member in one article and the computer exhibition he writes about in another. Although the articles authored by one particular journalist build a profile of this journalist, the meaning of their contents can be biased by the co-occurrence of the author's name. For the model whose purpose it is to depict semantic regularities of the written content, it thus makes sense to weigh parts

or paragraphs of a document less than other and sometimes even ignore other parts. As the effectiveness of these measures can better be investigated with more similar content, we simplify the procedure for our test corpus and discard any meta-information preliminary to training.

5.2.3 Stopword Removal

Much performance can be gained through removal of frequent tokens with little information value. Besides downsampling as proposed by Mikolov et al. [31] and implemented in the word2vec toolkit, it is possible to completely exclude a list of words. Independent of our test series for a variation of training parameters, we consider additional stopword removal to three different extents: removal of mere punctuation and quotes (1), removal of punctuation, quotes and, additionally, of articles and the compositional words und and $oder$ (2), and removal of punctuation, quotes and the full list of frequent German stopwords as contained in NLTK[13] (3), the standard library for natural language processing in Python. Additionally, we omit numeric values which, in the general case, do not carry any meaning without their concrete context.

5.2.4 Morphological Reduction

German is known as a language with considerably greater morphological richness than English. Additionally to verb tense, person and number, four grammatical cases and three genders determine the *declination* of nouns, adjectives, pronouns, articles and numerals. Contrary to English, verb forms differ in almost every *conjugation*. Tables 5.2 and 5.3 demonstrate the various inflexions of the German language. Although there are forms which are rarely used, each word class has several common morphological forms.

There are two basic options of how this diversity can be dealt with in a preprocessing step: *Stemming* and *lemmatisation* [29, chapter 4]. *Stemming* is the process of reducing words to their stem by stripping words off their language-specific affixes. While this can be achieved with relatively few rules, it also introduces several problems. For many verb forms, the different tenses cannot be mapped to the same stem, e.g. $liegen$ - lag (lie, lay), which would be stemmed to the two different tokens $lieg$ and lag due to their changing root vowel. On the other hand, the same stem can result from very different words, thus resulting in additional disambiguation problems, e.g. $Macht$ - $machen$ (power, do), which would both be stemmed to $mach$ when applying lower-casing, although the first is, in this case, a noun. In a dictionary, words can be looked up by their *lemma* or *lexeme*, the basic form of a word. Respectively, lemmatisation refers to the process of mapping inflected forms to their lemma. For the German language with its numerous inflexions, this could prove as helpful as it is costly. The additional preprocessing effort, however, could be compensated by a significant reduction of the vocabulary size, increasing actual training speed.

Köper et al. [22] report slight accuracy improvements on different similarity and analogy tasks for models trained on lemmas over models trained on the original inflected word forms. As we are mainly interested in semantic relations, the aggregation of different inflexions promises

[13] `http://www.nltk.org/howto/corpus.html#word-lists-and-lexicons` [accessed 30 May 2016]

	Indicative Mood		**Imperative Mood**
Person	**Present**	**Preterite**	**Imperative**
ich	schlaf**e**	schl**ie**f	
du	schl**äf**s**t**	schl**ie**f**st**	schlaf
er/sie/es	schl**äf**t	schl**ie**f	
wir	schlafen	schl**ie**fen	
ihr	schlaf**t**	schl**ie**f**t**	schlaf**t**
sie/Sie	schlafen	schl**ie**fen	

	Subjunctive Mood	
Person	**Present Subjunctive (Konjunktiv I)**	**Past Subjunctive (Konjunktiv II)**
ich	schlaf**e**	schl**ie**f**e**
du	schlaf**est**	schl**ie**f**est**
er/sie/es	schl**af**e	schl**ie**f**e**
wir	schlafen	schl**ie**fen
ihr	schlaf**et**	schl**ie**f**et**
sie/Sie	schlafen	schl**ie**fen

Composed Tenses	
Perfect	*Present 'haben'* + **ge**schlafen
Plusquamperfect	*Preterite 'haben'* + **ge**schlafen
Future I	*Present 'werden'* + schlafen
Future II	*Present 'werden'* + **ge**schlafen + haben
Present Subjunctive - Perfect	*Present Subjunctive 'haben'* + **ge**schlafen
Present Subjunctive - Future I	*Present Subjunctive 'werden'* + schlafen
Present Subjunctive - Future II	*Present Subjunctive 'werden'* + **ge**schlafen + haben
Past Subjunctive - Perfect	*Past Subjunctive 'haben'* + **ge**schlafen
Past Subjunctive - Future I	*Past Subjunctive 'werden'* + schlafen
Past Subjunctive - Future II	*Past Subjunctive 'werden'* + **ge**schlafen + haben

Table 5.2: Verb Conjugation. *As in English, standard verbs inflect into either weak or strong conjugation class. Additionally, there are irregular mixed forms. Listed above is an example conjugation of the strong verb 'schlafen' (to sleep). Changed vowels and added pre- and suffixes are marked in bold.*

Case	Singular m/f/n	Plural m/f/n
Nominative	der Hund	die Hund**e**
	die Maus	die M**ä**us**e**
	das gelb**e** Haus	die gelb**en** H**ä**us**er**
Genitive	des Hunds/**es**	der Hund**e**
	der Maus	der M**ä**us**e**
	des gelb**en** Haus**es**	der gelb**en** H**ä**us**er**
Dative	dem Hund	den Hund**en**
	der Maus	den M**ä**us**en**
	dem gelb**en** Haus	den gelb**en** H**ä**us**ern**
Accusative	den Hund	die Hund**e**
	die Maus	die M**ä**us**e**
	das gelb**e** Haus	die gelb**en** H**ä**us**er**

Table 5.3: Noun Declension. *Nouns inflect into one of four declension classes. Listed above are three example declensions of the nouns 'Maus' (mouse), 'Hund' (dog) and 'Haus' (house), the latter with the adjective 'gelb' (yellow), which also changes its ending depending on the noun's gender and number.*

multiple benefits: It increases the density of examples and it frees us from purely inflected forms when putting the model into use and looking up the most similar word representations for a given word. Additionally, reduced sparsity could improve our clustering results, either by enabling us to extract similarly detailed semantic information in less feature dimensions (one of the parameters in word2vec) or by leading to a more accurate distribution of word representations in vector space.

One negative aspect of normalisation in terms of lower-casing was already mentioned above: Despite their difference in meaning, some words will be mapped to the same normalised token. One reason tokens are lower-cased in many applications is that additional tokens can emerge as a result of sentence boundaries. Even if all sentence boundaries are detected correctly, which again requires rule-based (ignore common abbreviations) or more intelligent intervention (detect sentence boundaries with a syntactic tree parser), there are two possible further procedures. Firstly, all first words in a sentence can be lower-cased, causing the opposite problem that some actually proper nouns are misspelled. Secondly, words can be lower-cased or kept in their original form depending on which use is most probable. In many cases, there is only one correct spelling - the number of correctly preprocessed tokens can thus be increased, but at considerable cost. Beside the complication with sentence boundaries and truly unintentional misspellings, there exist, strictly speaking, various other sources for incorrect spelling. Common examples include capitalisation in headings, capitalisation for emphasis purposes or arbitrary upper and lower case for named entities. In sample checks we observed lower quality of word embeddings in the case of normalisation to lower case.

For complete training, we thus use unnormalised words[14], expecting misspellings and single occurrences to be generally reduced for the given editorial content.

5.2.5 Extraction of Multi-Word Concepts

Mikolov et al. [32] introduce phrase detection as an extension to their word embedding models and report improvements of the overall accuracy. In the English language, it is a special challenge to identify which words constitute one concept, i.e. belong to one *multi-word unit* (MWU). Although German has considerably less problems with MWUs as these are generally formed into one large compound noun, both the context of enterprise search and the news context contain personal, geographic and institutional names, concepts comprised from two or more words separated by whitespace. Ignoring these could result in a lower quality of the semantic regularities contained in the model. To find them, phrase detectors can be instantiated from the class `gensim.model.Phrases` before training the word2vec model. An instance of this class collects collocation counts for all pairs of single tokens within the passed list of text segments. Applied to the corpus before generating the training pairs, two tokens a and b are combined into a phrase (one token with a and b, joined by an arbitrary delimiter) if the following score for these tokens is higher than a specified threshold:

$$score(a, b) = \frac{(occ(a, b) - min\_count) \cdot |V|}{occ(a) \cdot occ(b)} \tag{5.1}$$

In equation 5.1, $occ(a, b)$ specifies the number of collocated occurrences of $a\,b$, min_count is an integer value free of choice defining the minimum required occurrences of the phrase in the detector input and $|V|$ is the size of the phrase vocabulary. The phrase vocabulary is not identical to the corpus vocabulary as built for the neural word embeddings, but contains all uni- and bigrams generated from the input to the phrase detector. The threshold should thus be chosen depending on the size and structure of the vocabulary.

5.2.6 Parameter Selection

Meaning and effect of parameters in implementations of the collected mechanisms in word2vec were analysed in section 3.5. We found the reasons for a certain choice of parameters in research publications often well hidden or without further notice, which may be due to uncertainties and empirical, not entirely systematic approaches. Although we cannot relate and justify each and every choice of parameter, we want to increase reproducibility by providing the key factors which restricted the set of configurations explored in section 5.3.

We set out with a standard configuration including the two variable parameters of model architecture and size. We justify the decision to experiment with a variable number of feature dimensions throughout all test series in two respects: Firstly, we observe the lack of a mathematical formula which would derive the best feature dimensionality between the minimum required differenciation and the threshold to overfitting from corpus and vocabulary size. Secondly, regarding the imminent clustering tasks and the omnipresent curse of dimensionality (see also section 4.2), it is not clear for us to say how much dimensionality reduction as part

[14] except common transliteration of German umlauts and 'ß'

Parameter	Description	Values
size	The number of feature dimensions the word is projected to	100, 200, 300*, 400*, 500*, 800
sg	The model architecture to use, either CBOW (0) or skip-gram (1)	0*, 1*
alpha	The adaptive learning rate	0.05*,**, 0.025*,**
seed	The seed for random initialisation of the M_{words} weight matrix	11***, 41***, 47***
window	The maximum distance of a context word to the focus word, to either side, uniformly sampled	2, 5*, 8
min_count	The minimum occurrence of a token in the corpus, for subsampling	5*
sample	The downsampling threshold, reduces the influence of frequent with assumed low information value	10^-4*, 10^-5, 10^-6
hs	Whether to use hierarchical softmax (1)	0*, 1
negative	The number of negative samples, if not using hierarchical softmax	5, 15*, 25
iter	The number of iterations over the corpus (also *training epochs*)	15*
workers	The number of kernels used in parallel, occasional lock-free overwriting of word vectors	8*, 64****

* used/compared over all test series
** 0.05 for CBOW, 0.025 for skip-gram, according to [33]
*** random choice, also used to shuffle documents
**** for lemmatised models (new hardware environment)

Table 5.4: Training Parameters. *Listed on the right are all test values of gensim training parameters selected for evaluation.*

of the context projection might help in achieving the best possible clustering results. Similarly, irrespective of the fact that the model architecture CBOW offers a considerable advantage regarding training time, we have not found consistent evidence that one excels over the other. More specifically, [28, 32] claim skip-gram is a better alternative than CBOW. [28] further present various scores where skip-gram performs better - not on all but on a predominant number of test collections. However, results in [4] imply the opposite. In [28], despite their empirical determination of skip-gram as the winner, Levy et al. emphasise the potential of CBOW. As our test series do not include a detailed evaluation of the learning rate, we follow the recommendation of Mikolov et al. in [32] and use values of 0.05 and 0.025 for CBOW and skip-gram, respectively. The choice of a higher learning rate for CBOW can be traced back to the fact that there is only a fraction of weight updates equal to the number of context words per training pair (see also section 3.5).

5.3 Evaluation Datasets

With multiple models trained in different parameter configurations, the question remains how their quality can be evaluated. For the English language, there exist a variety of evaluation test sets widely used among researchers. This is specifically important as they provide a baseline against which to make comparisons. While there are only few publications consistently using the same German test set and providing benchmarks, the database of paradigmatic semantic relations collected by Scheible and Schulte im Walde [52] has caught our particular attention. It provides a large collection with representative entries for word type and relation and annotations gathered in a crowdsourcing experiment on Amazon Mechanical Turk[15]. The mentioned shortcomings of German test sets for evaluation of distributed language models shall be addressed in two ways. On the one hand, the few results officially reported will be used for comparison to results of the trained models obtained with the respective test set. On the other hand, as quality is a function of how well models are suited for our particular task, we present another test set. For creation of the test set it is worth having a look at how others are created. In the following, a selection of commonly used evaluation data is analysed with respect to purpose and construction. Thereafter, we present the methods and motivation behind our evaluation strategy.

5.3.1 Measurement Quality Concerns

There are ongoing discussions about the value of syntactic versus semantic regularities evaluation [9, 22, 28, 54]. Words with syntactic (or rather grammatical) relation are also semantically related. Consider the example $good, better$: while $better$, from the grammatical perspective, is simply the comparative of $good$, this builds on the conceptual and semantic relation that two words linked by this are compared with regard to their degree of goodness. Within the same context, the two words $good$ and $better$ impose a relation on the two words compared. The main difference is that grammatical relationships can be handled to a certain extent by other techniques such as lemmatisation or stemming - but these have difficulties with grammatical irregularities which some try to avoid by applying additional rules. With these rules, common irregularities such as the comparative $better$ can be handled as an exception. Word embeddings on the contrary build on the mere co-occurrence of words, decoupled from their sequential order and encoded within their feature dimensions, and by this nature make no difference between regular and irregular forms. The chances of $better$ having a vector similar to $good$ are the same as for big to have a vector similar to $bigger$, which are also related in spelling.

5.3.2 Semantic Similarities

A good first indicator of how well semantic regularities are captured in a model is whether the representations of synonymous or closely semantically related words have similar characteristics.

[15] platform for coordination of requesters and workers, which can complete tasks of the former voluntarily or for monetary compensation, available at `https://www.mturk.com/mturk/welcome` [accessed 15 July 2016]

For distributed word representations, similarity is measured by the cosine of the angle between two respective word vectors (see chapter 3) - the higher the cosine, the higher the similarity.

One test set to evaluate this was originally proposed for the English language by Rubenstein and Goodenough [47] and contains 65 word pairs with averaged similarity judgements from 51 participating students, ranging from unrelated to synonymous on a scale from 0.0 to 4.0. Before assigning scores, participants were asked to sort the word pairs according to their similarity. A German translation, denoted in the following as GUR65, was published by Gurevych [13].

A second test set for evaluation of semantic relatedness in English language models was proposed by Finkelstein et al. [10]: The *WordSimilarity-353 Test Collection* is composed of 353 word pairs together with human-assigned similarity judgements. Similarity scores were averaged over all subjects and their estimation of relatedness on a scale from 0 to 10, with 10 implying maximum relatedness. The word pairs were later reassigned by Agirre et al. [1] to one set focused on measuring similarity and another focusing on relatedness. Schmidt et al. [53] translated the original test collection into German with the help of several volunteers and three different online dictionaries. Word pairs with less than a two-thirds agreement on a single translation were discarded to ensure transferability of concepts, resulting in 280 word pairs and their assumed language-independent ratings. Similarity judgements were reconducted to eliminate potential discrepancies by Köper et al. [22], following the same procedure, and subsequently applied to evaluation of language models. We subsequently refer to this version as WORDSIM280.

A different approach to evaluate the preservation of semantic regularities in terms of synonym identification is applied by Landauer and Dumais [24]: 80 multiple-choice questions from the original *Test of English as a Foreign Language* (TOEFL) are used to compare the performance of language models to that of human test-takers in correctly choosing the one synonym in each four possible answers. Mohammad et al. [38] suggest a corresponding set of 1008 closed similarity questions with four alternative answers originating from a word choice quiz of the German edition of the magazine *Reader's Digest*. Following the lines of Köper et al. [22], we disregard questions containing multi-word expressions and refer to the resulting subset of 426 questions as RD426.

5.3.3 Regularities Expressed by Analogies

A second category of evaluation datasets comprises those constructed to evaluate how well language models capture semantic and syntactic regularities. Mikolov et al. [33] introduce the *MSR Analogy Test Set* (*MSR*) within the scope of a Microsoft Research funding. It contains 8000 analogy pairs emphasising mostly syntactic relationships, grouped into different categories such as *singular - plural noun*, *singular 's-Genitive*, *comparative - superlative* or *infinitive - past tense*.

The *Google Analogy Test Set* (*Google*) is provided by Mikolov et al. [31] together with the introduction of the toolkit word2vec. Compared to MSR it contains 9128 analogy pairs, evaluating the preservation of regularities of not only syntactic but also semantic nature. Results have been reported in many publications [2,4,8,22,26,28,31,33,36,42], where semantic evaluation commonly refers to analogy tasks testing the following connections: Countries and

their capitals, states and each their big cities, monetary unions and their currency as well as family relationships targeted at evaluation of the gender regularity, e.g. *brother to sister as father to ? (mother)*. For our model evaluations, we use a German translation of the test set as provided by Köper et al. [22], disregarding the grammatical section *adjective - adverb* which mostly translates to the identical word. We refer to the resulting subset of 18 552 questions as GOOGLE18552.

The *Paradigmatic Semantic Relation Analogy Test Set* was constructed by Köper et al. [22] from the database of paradigmatic semantic relations explicitly for the German language with annotations gathered from crowdsourcing by Scheible and Schulte im Walde [52]. Scheible and Schulte im Walde created the database in two steps: For the first step, a representative distribution of target words was extracted from *GermaNet*, the German equivalent to *WordNet*[16]. Suggestions of each an antonym, a synonym and a hypernym were collected from 10 human participants for either target word. In a second step, 10 participants were asked to rate the relation strength of the proposed pairs for each of the three relation types, independent of the one proposed before. Köper et al. select all word pairs suggested in respect to a relation by at least 4 out of 10 participants and consider both combinations of each pairs *a* and *b* as analogy question for the same relation and word type (noun, adjective, verb). This results in a selection of 2 462 analogies, divided into the five sections *Adjective Antonym*, *Noun Antonym*, *Noun Hyponym*, *Noun Synonym* and *Verb Antonym*, subsequently denoted as SEMPARA2462.

5.3.4 Construction of a Representative Test Set for Evaluation of Paradigmatic Relations

In the last paragraph we outlined the construction of SEMPARA2462. During closer examination in the course of our experiments, we perceived the general validity of the associated relations as questionable. To illustrate this position, several relation pairs participating in respective analogy tasks are listed in table 5.5.

Based on this observation, we decided to extract a second analogy test from the database, which, with a total of 1 684 target-response pairs rated for strength of relation, offers a great extent of potential analogy tasks. For construction of our test set, we determined the following subgoals:

1. supplement SEMPARA2462 in terms of task sections for *Adjective Hyponym*, *Verb Hyponym*, *Adjective Synonym* and *Verb Synonym*

2. fulfil the requirement for stable results per section with an appropriate number of tasks per section

3. facilitate a fair evaluation of all analogy pairs by randomising which participating word is to be predicted

After review of the ratings per relation in the database of paradigmatic relations, we selected all pairs rated with an average score of 4 or higher for their strength of the respective relation.

[16] lexical database with interlinks between conceptual-semantically and lexically related sets of synonyms (Synsets)

Aiming at a selection of commonly known and unambiguous representatives for each relation, we restricted our choice to the word pairs actually rated by all 10 participants (no omissions).

Target Word	**Suggested Antonym**	**Criticism**
neugotisch	altgotisch	*Neu* (new) is antonym to *alt* (old), but *neugotisch* (neo-Gothic) and *altgotisch* (classic Gothic) mainly describe two of various architectural styles, is there anything to be considered a true antonym?
süß	bitter	There are 4 commonly recognised basic tastes: sweet, sour, salty and bitter, more recently complemented by a fifth, 'umami' - how is *süß* (sweet) the antonym of *bitter* (bitter)?
grün	rot	In some context such as that of traffic lights, *grün* (green) and *rot* (red) can be considered antonyms, but in the general sense, how can a color be the antonym of another, apart from possibly black (absence of color) and white (almost infinite light waves of different lengths)?
Breite	Länge	What about more dimensions i.e. *Höhe* (height) as the common third dimension to *Breite* (width) and *Länge* (length)?
VHS	DVD	What characterises modern DVDs as the antonym to VHS, would it not be better defined as its successor?
Bibel	Koran	Bible and Quran are both central religious texts, the former of Christianity, the latter of Islam, does this not rather emphasise a parallel than making it antonyms of one another in a general context?

Table 5.5: Example Word Pairs of SemPara2462. *Excerpts from* SemPara2462 *demonstrate that the representation of respective relations in some word pairs are questionable.*

In a qualitative assessment, we observed that, frequently, models predict one participant of an analogy correctly, but fail on another, implying the relation is not equally clear for each member of the analogy. This can be attributed to multiple reasons: Firstly, the word pairs can be connotated, e.g. we observed the task $good\,to\,bad\,as\,light\,to\,?\,(dark)$ being solved correctly more often than the incomplete analogy $bad\,to\,good\,as\,light\,to\,?\,(dark)$. The relation vector can be imagined to point into the opposite direction. Due to high dimensionality of the feature space, resulting best possible predictions are sometimes words similar to the target word and sometimes completely unrelated words - depending on the semantic length of the

relation vector. Secondly, the target word of the pair to be complemented, in the above case $light$, can be ambiguous, i.e. have different meanings and/or represent multiple lexical entities ($light$, the noun vs. $to\,light$, the verb). Thirdly, some of the participating words or their relation can be under-represented in the training corpus. Only as a final inference, the model can be assumed to preserve semantic relationships in language insufficiently, either due to an unfavourable choice of parameters or an unlucky initialisation of weights (local optimum for gradient descent optimisation). To meet the third subgoal, we randomised the four-word combination forming the analogy pair. In contrast to Köper et al. [22], we include one combination of two word pairs but randomise the choice of either *pair1 pair2* or *pair2 pair1*, as well as the pair direction of either *word1 word2* or *word2 word1*, except for the case of directed hyponym relation, where we use the originally suggested and rated pair direction. The hyponym sections thus contain each one randomised combination of pairs with the hyponym as the target word and the suggested word for the second pair, i.e. the more general term, as the word to be predicted. The resulting test set contains task sections for all nine possible combinations of word class and relation type from the database of paradigmatic semantic relations [52] with a minimum of 406 tasks for section *Noun Antonym* and 1867 on average. We subsequently refer to the test set of 16808 analogy tasks as ComPara16808.

5.3.5 Metrics

For Gur65 and WordSim280, we compute two values to measure correlation between human-annotated similarities and cosine similarities per model, namely Pearson's correlation coefficient and Spearman's rank correlation coefficient [55]. Pearson's sample correlation coefficient, or *Pearson's r*, rates the linear dependence between two datasets of size n $\{x_1, ..., x_n\}$ and $\{y_1, ..., y_n\}$ on a scale of -1 to 1 with 1 indicating a total positive correlation, 0 no correlation and -1 total negative correlation. Independently from measurement units, the correlation coefficient assesses to what extent the respective values of two variables X and Y arrange themselves into an order with similar (linear) distances between one another. More specifically, the correlation coefficient indicates whether the distance from each x_i to x_{i+1} is similar to the one from y_i to y_{i+1} in proportion to the respective range of values. Applied to word pairs, we test whether the human-annotated similarity scores correlate well on a linear scale with the cosine similarities computed from a model.

Spearman's rank correlation coefficient or *Spearman's rho* measures the statistical dependence between two rankings. Unlike Pearson's r, Spearman's rho only describes to what extent $\{x_1, ..., x_n\}$ and $\{y_1, ..., y_n\}$ arrange themselves into a similar order. The rank correlation is likewise expressed on a scale of -1 to 1, with 1 indicating the exact same ranking and -1 indicating a reverse order, e.g. in case of similarity rankings to cat $\{kitty, pet, dog\}$ versus $\{dog, pet, kitty\}$. Thus, a model can have a good Spearman value for a similarity test set, although there is no consistent proportion between the two complete sets of similarities (lower Pearson correlation). We expect the Pearson correlation to be a better indicator of how well linguistic regularities are encoded within a model. As most reports, however, refer to Spearman's rho [8, 22, 28, 31, 33, 42], we compute both.

For RD426, Google18552, SemPara2462 and ComPara16808 we extended *gensim's* method `gensim.models.Word2Vec.accuracy` to accept an additional lambda function

to be applied to each test case line. This enables us to choose a comparison of lower-cased or otherwise transliterated words independent of the original representation in the test file. The accuracy score signifies the percentage how many of all successfully performed analogy tasks were completed correctly. To solve the analogy tasks, we use the multiplicative cosine *3CosMul* introduced by Levy and Goldberg [26] (see also section 3.5.5), which a few sample checks manifested itself as the better alternative to the originally proposed *3CosAdd* [33].

5.4 Discussion

Our evaluation comprises 197 models. We decided to explore specific parameters based on 6 default configurations (in two random initialisations), marked with single asterisk in table 5.4. With variations in model architecture, size and initialisation, this resulted in 2 x 3 x 2 = 12 models, serving as statistical foundation for our test series. We ran five test series to evaluate the influence of the following parameters on CBOW versus skip-gram:

1. projection size
2. window size
3. downsampling rate
4. hierarchical softmax and different numbers of negative samples

For each parameter variation from the test series we trained and evaluated 12 models as stated above, with the exception of projection sizes which we completed with values of size 100, 200 and 800 for CBOW and skip-gram in otherwise default configuration. Resulting, we have 144 models for these tests. Additionally, we conducted a few tests with varying stopword removal, and out of specific interest e.g. in effects of even higher window size and of fewer iterations. Our models with 10 compared to 15 iterations indicated that 5 additional iterations indeed help to improve quality for this particular corpus - we excluded the former from evaluations. Scatter plot visualisations per test score and differentiated presentation of each parameter variation can be found in the appendix on page 89. Hereafter, we discuss individual aspects in respect to score variance, maximum, minimum and average scores per test (subset) and, finally, which parameters influence certain model capabilities. Overall test scores are listed in table 5.6 (for separate scores of CBOW and skip-gram, see table 5.7).

A closer inspection of score variance across test sets and sections reveals an especially high variance of 41.2% for the semantic tasks of GOOGLE18552. The highest score with 66.6% accuracy is reached by two skip-gram models with window sizes of 8 and 10. This is remarkable in that for both window sizes, only a small number of models was trained at all. The semantic task set is composed of the following sections: *capital-common-countries, capital-world, currency, city-in-state* and *family*. As such, a vast majority of the accuracy tasks aims at geographic relations. One possible reason for low scores in this subset of GOOGLE18552 is that the geographic relation between relatively unknown countries or states and their cities and currencies is not well represented in the training corpus. Here, CBOW, whose weights are only updated once per training example, has a competitive disadvantage. This shows in a variance of over 40% where the the best and worst skip-gram model differ in just below

	Gur65		WordSim280		RD426	Google18552	
	rho	r	rho	r	total	sem	synt
Median	0.70	0.69	0.69	0.65	77.6%	55.9%	47.5%
Mean	0.70	0.69	0.69	0.65	76.9%	53.8%	46.6%
Min	0.59	0.59	0.55	0.54	65.6%	25.4%	36.6%
Max	0.82	0.78	0.75	0.69	82.3%	66.6%	51.1%

(a) Overall - Similarity Correlations + Accuracies

		noun	adj	verb	ant	hyp	syn	total
SemPara 2462	**Median**	7.4%	8.1%	0.8%	6.7%	0.7%	10.0%	7.0%
	Mean	7.3%	8.3%	1.0%	7.2%	0.9%	10.5%	7.0%
	Min	3.6%	3.2%	0.0%	2.8%	0.0%	2.4%	3.2%
	Max	14.1%	13.8%	2.5%	12.2%	3.3%	20.5%	11.2%
ComPara 16808	**Median**	3.60%	3.40%	3.10%	2.70%	2.30%	4.50%	3.40%
	Mean	3.46%	3.18%	3.06%	2.66%	2.25%	4.27%	3.20%
	Min	1.00%	1.10%	1.00%	0.80%	0.70%	1.30%	1.20%
	Max	5.20%	4.30%	4.60%	3.90%	3.40%	6.20%	4.50%

(b) Overall - Accuracies SemPara2462 and ComPara16808

Table 5.6: Overall Scores of Word Embeddings

30%. Similarly, the effect of higher window size seems to be beneficial for rare words, which accordingly become part of a training context more often. While this generally applies to all words, by subsampling, downsampling and adaptive learning rate, rare words could benefit the most from this higher maximum context size. On the other hand, this effect can be expected to be less stable due to randomised context sampling.

Results for SemPara2462 are generally low (see table 5.6b). The only benchmarks available for this test set are the ones reported by Köper et al. [22] for each word-based and lemma-based models trained with CBOW, skip-gram and the standard statistical approach of positive point-wise mutual information weighting and subsequent singular value decomposition. The six model scores range from a minimum of 13.8% (lemma-based skip-gram) to 15.1% (lemma-based CBOW) but relate to a subsequently described recall at 10. The scores for CBOW on subsets of SemPara2462, reported separately, range from 0% for the word-based CBOW model on antonym tasks to 8.6% for the lemma-based CBOW model on the synonym subset.

We can confirm the effect of proportionally high scores on the synonym subset with our evaluation results on SemPara2462. While the scores for other subsets reach a maximum of only 12.2% (antonym subset), CBOW reaches 20.5% on the synonym subset. Among the top 10 models in regard to the subset of synonym tasks, there are only two skip-gram models at positions 6 and 10. Also noteworthy is that both skip-gram representatives and 7 out of the 10 models were trained with hierarchical softmax. However, the worst CBOW model reaches no

		Gur65		WordSim280		RD426	Google18552	
		rho	r	rho	r	total	sem	synt
CBOW	Median	0.68	0.68	0.69	0.65	77.0%	52.8%	48.3%
	Mean	0.68	0.68	0.68	0.65	76.3%	49.5%	46.7%
	Min	0.59	0.61	0.64	0.58	65.6%	25.4%	36.6%
	Max	0.75	0.74	0.71	0.68	81.8%	59.3%	51.0%
Skip-Gram	Median	0.73	0.72	0.72	0.66	78.0%	60.4%	46.7%
	Mean	0.72	0.71	0.70	0.65	77.5%	58.2%	46.6%
	Min	0.65	0.66	0.55	0.54	71.2%	42.8%	37.3%
	Max	0.82	0.78	0.75	0.69	82.3%	66.6%	51.1%

(a) CBOW vs Skip-Gram - Similarity Correlations + Accuracies

		SemPara2462						
		noun	adj	verb	ant	hyp	syn	total
CBOW	Median	7.5%	6.7%	1.2%	5.8%	1.1%	13.1%	6.5%
	Mean	7.1%	6.7%	1.2%	5.7%	1.1%	12.4%	6.3%
	Min	3.6%	3.2%	0.0%	2.8%	0.0%	2.9%	3.2%
	Max	10.0%	9.5%	2.5%	7.8%	3.3%	20.5%	8.4%
Skip-Gram	Median	7.3%	10.0%	0.8%	8.7%	0.4%	8.2%	7.8%
	Mean	7.5%	9.9%	0.8%	8.6%	0.6%	8.5%	7.6%
	Min	4.3%	3.2%	0.0%	3.5%	0.0%	2.4%	3.6%
	Max	14.1%	13.8%	2.5%	12.2%	2.2%	17.6%	11.2%

(b) CBOW vs Skip-Gram - Accuracies SemPara2462

		ComPara16808						
		noun	adj	verb	ant	hyp	syn	total
CBOW	Median	3.70%	3.25%	3.70%	2.60%	2.60%	4.70%	3.40%
	Mean	3.54%	3.04%	3.46%	2.48%	2.45%	4.48%	3.29%
	Min	1.00%	1.40%	1.00%	1.00%	0.70%	1.30%	1.30%
	Max	5.20%	4.20%	4.60%	3.50%	3.40%	6.20%	4.50%
Skip-Gram	Median	3.60%	3.60%	2.70%	3.00%	2.10%	4.30%	3.30%
	Mean	3.38%	3.32%	2.65%	2.84%	2.03%	4.05%	3.10%
	Min	1.20%	1.10%	1.30%	0.80%	1.00%	1.30%	1.20%
	Max	4.50%	4.30%	4.00%	3.90%	2.80%	5.30%	3.90%

(c) CBOW vs Skip-Gram - Accuracies ComPara16808

Table 5.7: Comparison of Test Scores for CBOW and Skip-Gram Models

more than 2.9% on the same tasks. The high variance can be explained by the (intentionally) small vector offset which describes the synonym relation. The unstable result emphasises the fact that the capabilities of a model to find synonyms cannot be evaluated on analogy tasks.

In contrast to our expectations, the scores for ComPara16808 are lower than for Sem-Para2462 (see table 5.6b).

For the semantic subset of Google18552, Köper et al. [22] report accuracies for German CBOW and skip-gram models trained on a large web corpus with 42.4% and 45.9%, respectively. Evaluation of the lemma-based version shows an increase of the former to 43.5%. In the same experiment, two English models were trained, for which results exceed the ones of the German models by a considerable margin of about 25%. Baroni et al. [4] compare results across several English language models, among them 48 CBOW models, as well as 38 state-of-the-art statistical models with word mappings to both full and reduced context vectors, including two publicly available models. In their evaluation, the best CBOW model performs best overall with 66% on the semantic subset.

The maximum accuracy achieved for the syntactic subset of Google18552 with our models based on the German *Spiegel* corpus is 51,1%. Compared to the other subset, variance is low with 14,5%, the average 46.6%. The syntactic sections of Google18552 evaluate the linearity of grammatical relations such as country and nationality adjective, different tenses, comparative forms, but also include 812 analogy tasks with opposite adjectives.

As distributed word embeddings treat each token as an atomic semantic unit, they make no difference between whether words share the same stem. Thus, opposite adjectives as contained in the respective section of Google18552, e.g. $produktiv - unproduktiv$ (productive - unproductive) are processed just the same as antonyms of Sem-Para2462, e.g. $arm - reich$ (poor - rich).

Due to the scope of this thesis and the limited time horizon we could not complete our evaluations in time for further investigation of lemmatised models and the extension of the vocabulary to recognised multi-word units (phrases). Alongside our general evaluation, we prepared a lemmatised corpus using the additional library *mate-tools*[17] and conducted 17 trainings with different parameter combinations.

For our clustering analysis we are going to analyse three models: one skip-gram model, one CBOW model and one of the models trained on the lemmatised *Spiegel* corpus. Following the advice for a task-specific evaluation in [9, 54], i.e. to take account of which test values are significant to the imminent task, we base our decision on a respective subset.

We approach the broad restriction by filtering models which lie above the average for both the similarity test sets, Gur65 and WordSim280, and the multiple-choice test RD426. According to our expressed concerns about mere ranking correlation constituting a good decision criterion, we consider the linear Pearson correlation coefficient for the former. We then take the model with respective best result on the analogy questions for the semantic subset of Google18552. With the generally minor performance on both SemPara2462 and ComPara16808 we do not want to risk selecting a model which was lucky by chance but might not be a good choice considering more than the closest word vector as correct or, respectively, incorrect prediction of the missing word. A comparison of reported results by Köper et al. [22] shows that test values for SemPara2462 improve by up to 21.4% for the case of synonymous

[17] available at `https://code.google.com/archive/p/mate-tools/` [accessed 20 July 2016]

nouns when adapting evaluation to this problem. They tested once for the correct solution being the exact most similar word applying the multiplicative cosine *3CosMul* and once for the correct solution being among the top 10 most similar words, same method applied. Our resulting selection and the respective parameterisation is listed in table 5.8.

	model	**alpha**	**size**	**win**	**min**	**sample**	**neg**
SKIP	skip-gram	0.025	500	8	5	10^{-4}	15
CBOW	CBOW	0.05	400	10	5	10^{-4}	15
LEMMAC	CBOW	0.05	400	5	5	10^{-4}	25

Table 5.8: Selection of Models. *One skip-gram, one CBOW and a lemma-based model were selected for clustering analysis with respect to their performance on evaluated test scores.*

6 Evaluation of Semantic Clustering on Word Embeddings

Clustering of word representations was formulated as a major goal within the title of this work. Findings during initial sample inspections indicated the urgent need of preliminary model selection in accordance with the task and a more educated choice of clustering algorithms. In chapter 4, we approached the latter with an analysis of applicable clustering algorithms. Chapter 5 constituted the last step towards the final evaluation of clusterings. Although the extrinsic evaluation of distributed word representations in section 5.4 could not give a definite answer as to which training configuration is best suited for this task, we chose one skip-gram model and two CBOW models, one lemmatised, with best performance on a relevant subset of the computed benchmarks. Hereafter, we conduct a qualitative evaluation. Section 6.1 documents the experimental setup, subsequently discussed in section 6.2.

6.1 Qualitative Evaluation

The evaluation of embeddings in chapter 5 revealed how difficult it is to make a profound decision based on common benchmarks. Besides a possible application to information retrieval (IR) in the context of enterprise search, we take cluster analysis as a means of qualitative evaluation. If the terms within clusters share clear characteristics and the clusters are well distinguishable from one another, this indicates a good model on the one hand and, on the other hand, a potentially useful grouping for the application to IR. We are mainly interested in two things: a good representation of similarities and the encoded relation of abstract concepts and their concrete manifestations. The former can facilitate semantic query extension by finding also documents that do not contain the exact query terms but a semantic synonym. The relation of hypernyms and hyponyms is relevant insofar that we want to find out whether it is in any way feasible to extract a local taxonomy as motivated in the introduction. Also, we hope to gain additional insight into why the test scores for the respective subset of tasks from SemPara2462 and ComPara16808 were so low. For the analysis, we consider the following terms:

- *Tor* (gate, goal)
- *Getränk* (beverage)
- *Tee* (tea)
- *Pfefferminztee* (peppermint tea)

- *Früchtetee* (fruit tea)

Tor constitutes a good example of an ambiguous term. In the context of sports, it is the German word for *goal*, although it does not share the second English meaning as general objective. In the architectural context, it is the German word for *gate*, commonly used either for imposing entrance portals, garden gates or garage doors. The other terms construct part of a local hierarchy, illustrated in figure 6.1. With this hierarchy in mind we are going to analyse the different relations with each model's most similar vectors.

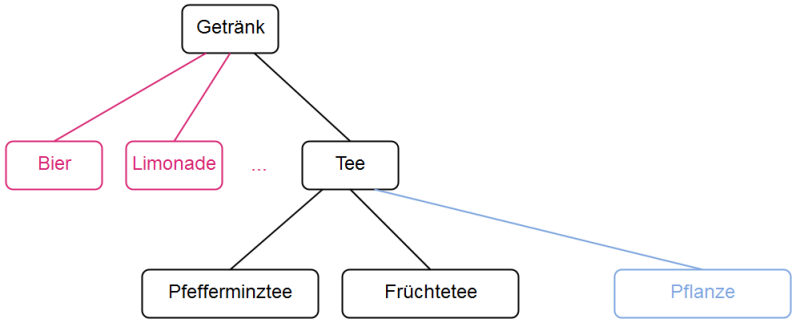


Figure 6.1: Local Taxonomy of the Hypernym *Getränk* (beverage)

We compute cosine similarities to retrieve the top 10 most similar terms for each item in the list.Table 6.1 lists the resulting terms per target for both CBOW and skip-gram. A comparison of CBOW to skip-gram reveals substantial differences regarding the type of relatedness towards the target which dominates the most similar word representations. For the target word *Tor*, CBOW yields several inflected forms as closest to the target. This actually implies a good semantic understanding of the word - if we worked with a lemmatised corpus, all inflexions would be mapped to one and would not interfere. The word *Tür* (door) as the first diverging term is an appropriate synonym regarding its first word sense. *Eingangstor* (entrance gate) is a German compound and poses a more fine-grained definition or hyponym. *1:0* clearly relates to an interpretation of *Tor* as a goal in sports, as well as *Siegtor* (winning goal) and *Stadion* (stadium).

Skip-grams most similar terms range from *Brandenburger (Tor)* (landmark of the German capital) to *Anschlusstreffer* (goal which leaves one team only one behind the other) with the inflected forms more evenly distributed over the top 15. More difficult to interpret are terms such as *Uranias* (Hamburg sports club), *faustet* (blocking the ball with a fist) or *Granitza* (retired German football player), which apparently refer to the sports meaning, but would be expected to have less weight in the observation. It is unclear why this particular sports club or football player are so close to the target word and why others are not.

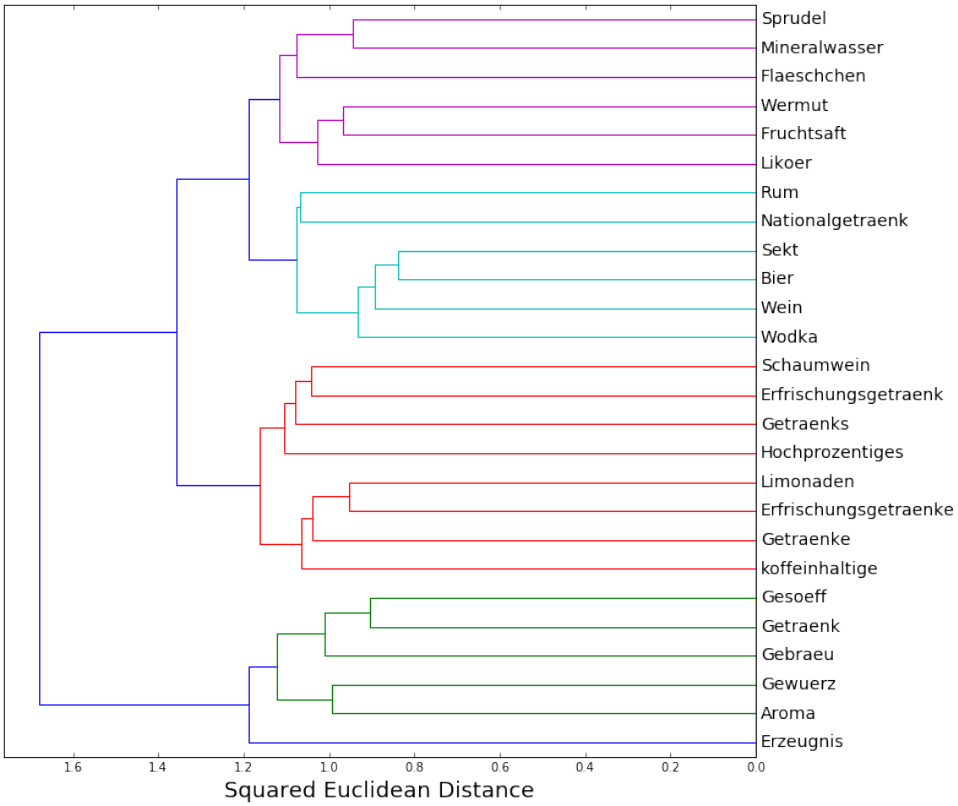
For the words *Tee, Pfefferminztee* and *Früchtetee*, CBOW suggests the respective alternatives commonly enjoyed for breakfast. *Kaffee* (coffee) is always first, other suggestions include *Mokka* (mocha), *Frühstück* (breakfast) or *Tasse* (cup): alternative beverages, the occasion to consume a (hot) beverage and the thing to drink from, most often used as a quantifier. At first glance, similar groups can be found in skip-gram as well, for *Früchtetee* (fruit tea) however, the model seems to be a bit unlucky. The top suggestions align not so well with the intuition: what does relate fruit tea closely with salami sandwiches, vegetable juice, tuna cans or ox tongue other than its edibility?

	CBOW	Skip-Gram
Tor	**Tore, Tores, Tors, Tür, Eingangstor, 1:0, Fenster, Siegtor, Pforte, Stadion**, Schlagbaum, Spielfeld, Eisentor, Strafraum, Haupttor, Ball, Hauptportal, Gittertor, Schlusspfiff, Flügeltür, Holztor, Glastür, Wachhäuschen, Torlinie, Eingang	**Brandenburger, Tore, Tores, Querlatte, Fan-Meile, faustet, Siegtreffer, Wiederanpfiff, Hallesches, Anschlusstreffer**, Uranias, Siegtor, Tür, Ball, Tors, Abseitstor, Pfostenschuss, Anstoßkreis, Körpertäuschung, abpfiff, Torwart, Blechtor, Strafraum, 0:3-Rückstand, Granitza
Getränk	Gesöff, Fruchtsaft, Wein, Erfrischungsgetränke, Bier, Erfrischungsgetränk, Nationalgetränk, Gebräu, Limonaden, Mineralwasser	Bier, alkoholisches, Cider, koffeinhaltige, Wein, grusinischem, Mixgetränk, Wodka, Sekt, getrunken
Tee	Kaffee, Pfefferminztee, Mokka, Frühstück, Tasse, Orangensaft, Milchkaffee, Cappuccino, Milchtee, Keksen	Kaffee, Tasse, Milchtee, trinken, ungesüßten, Kandis, aufgießen, gezuckertem, Cashewnüsse, brühte
Pfefferminztee	Kaffee, Tee, Mokka, gesüßten, Tasse, Gebäck, Fruchtsaft, Milchkaffee, Wermut, Früchtetee	Tee, Milchtee, Tomatensalat, ungesüßten, Kaffee, Plow, Hackbällchen, Minztee, gezuckertem, eingeweichte
Früchtetee	Kaffee, Tee, Milchkaffee, Pfefferminztee, Cappuccino, Apfelschorle, Rotwein, Thermoskanne, Instantkaffee, Eiskaffee	Salamibrötchen, Gemüsesaft, Kandis, ungesüßten, eingeweichte, Rehnüsschen, Thunfischdosen, Ochsenzunge, Gemüsesalat, aufbrüht

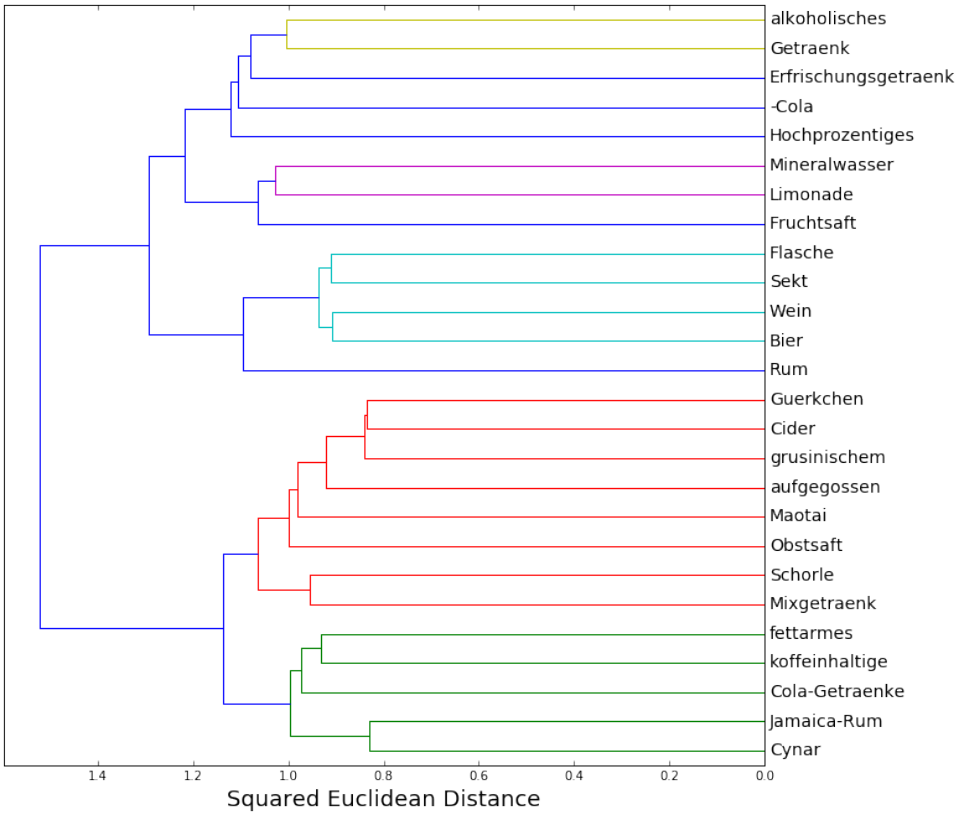
Table 6.1: Most Similar Terms for CBOW and Skip-Gram. *Listed are the terms with highest cosine similarity towards the respective target, computed from representations in each the selected CBOW and skip-gram model (p. 49).*

Apparently, skip-gram has more outliers due to its separate updates per single word pairs in contrast to CBOWs single update on an averaged context. Another possible inference is that it could still require additional training or the learning rate has to be chosen more carefully.

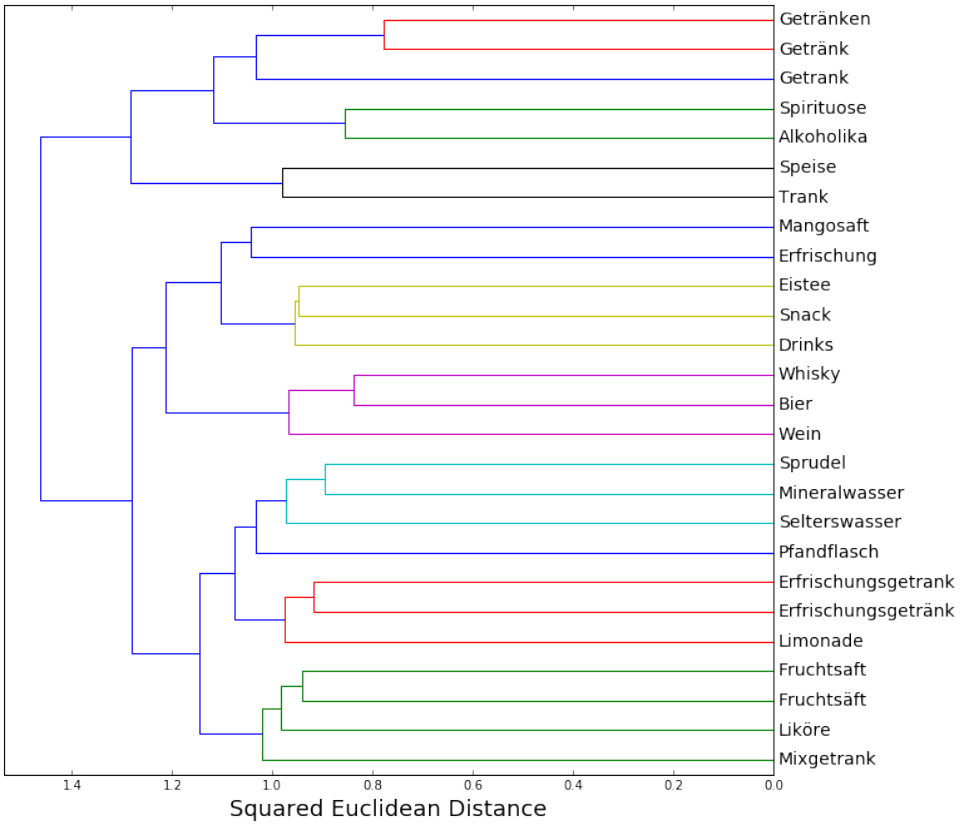
Words can have one to many different meanings, and these meanings can be either totally independent, as in the case of the disambigous term *Tor*, or they can be closely related. For this purpose, we prefer not to determine the number of clusters in advance, which we think should be dependent on the variance of meaning. Secondly, we presented different methods for the computation of group distances in chapter 4 and explained the problems methods such as *single linkage* or *complete linkage* have with outliers. As we could already observe outliers in the previous table of most similar terms, we choose to use Ward distance as a more sensitive choice in this respect and simply conduct agglomerative hierarchical clustering.



(a) Example Clustering CBOW



(b) Example Clustering SKIP



(c) Example Clustering LemmaC

Figure 6.2: Dendrogram for *Getränk*. *Agglomerative hierarchical clustering, Ward algorithm*

	CBOW	**Skip-Gram**
Getränk - Tee	0.40	0.38
Getränk - Pfefferminztee	0.38	0.26
Getränk - Früchtetee	0.32	0.29
Tee - Pfefferminztee	0.59	0.44
Tee - Früchtetee	0.51	0.48
Pfefferminztee - Früchtetee	0.49	0.41

Table 6.2: Similarity between Terms. *Listed are the cosine similarities for respective terms.*

6.2 Discussion

Next, we look at the dendrograms of some of our clusterings. We consider the dendrograms of all three models CBOW, SKIP and LEMMAC for the word *Getränk* (beverage), depicted in figures 6.2a to 6.2c. The purple cluster of CBOW consists of words related to water as a beverage, i.e. *fizz* and *mineral water*. Vermouth as a bitter wine often drunk as aperitif does not fit in well, we would hope for a clear separation between alcoholic beverages and water. Also, *Schaumwein* (sparkling wine) is grouped with *Erfrischungsgetränk* (refreshment) and *Hochprozentiges* (high-proof alcohol) instead of with its exact synonym *Sekt*, a not very obvious semantic relation.

The top cluster for the skip-gram model is a good grouping, as *alkoholisches Getränk* (alcoholic drink) does often occur as a phrase. On the other hand, *Getränk* as our point of departure is better grouped in CBOW with synonymous words such as *Gesöff* and *Gebräu*, which roughly corresponds to *brew* or *plonk*, i.e. a colloquial synonym. The problem we had previously observed among the top 10 becomes evident once more within the red cluster: *grusinischem* and *aufgegossen* bear no close relationship from first intuition.

Figure 6.2c depicts the clustering results with the lemmatised CBOW model. These are in fact quite impressive: The word forms of the same concept remaining after lemmatisation are closely grouped together, the upper green cluster represents synonyms of each other. *Speise und Trank* are a common expression. *Whisky*, *Bier* (beer) and *Wein* (wine) fit together from a broad perspective. Although the lower green cluster contains an alcoholic beverage (liqueur), fruit juice in two forms and *Mixgetränk* (mixed drink), these do fit together reasoning with their mutually attributed sweetness.

Recalling our local taxonomy (fig. 6.1), we were hoping for a substantially complete representation of typical beverages. Unfortunately, our chosen example beverage type, *tea*, is not among any 25 most similar terms. On the one hand, this suggests the lists do not end at this point. However, we have no rule of how to determine at which point the similar words become permeated with a considerable amount of seemingly unrelated words. On the other hand, we cannot observe the strength of representation directly from the dendrograms.

One way to visualise the word frequency in a corpus is to define the font size in proportion to the word count. As this needs a considerable amount of space, we decided create word clouds from our clustering. The respective word cloud for the hierarchical clustering from figure 6.2c

Figure 6.3: Word Cloud for *Getränk* (beverage). *Produced from* LemmaC, *corresponding to dendrogram in fig. 6.2c*

is depicted in figure 6.3.

From various options of feature transformation we observed the best results in two- or three-dimensional space with randomised PCA (see `http://scikit-learn.org/stable/modules/decomposition.html` [accessed 24 July 2016]): Terms belonging to the same word type are grouped relatively close together A scatter plot for the most similar terms to *Getränk* is depicted in figure 6.4.

We omit other visualisations such as the two or three first principal components derived from classical PCA and t-SNE[1] as these did not offer any significant insight.

As a final experiment, we want to compare our previous cluster results with ward clustering on vectors preliminary reduced with randomised PCA. An empirical estimate shows 5 is a good value. The resulting dendrogram is depicted in figure 6.5.

6.3 Summary

After our qualitative analysis we can confirm that hypernyms are very rarely among the top most similar candidates for a term. This can be explained by the fact that, the further up the hierarchy of a subconcept, the more meaning is contained in an overarching term which is not related to the subconcept itself. Hence, to retrieve the hypernym from a subconcept without any additional context information is infeasible from the current state of knowledge. A second noticeable conclusion is that, as well as neither the top most similar terms nor the following grouping can be expected to be free from errors, skip-gram seems to contain much

[1] Algorithm based on nearest neighbour-search, for visualisation of high-dimensional data (see `http://scikit-learn.org/stable/modules/manifold.html` [accessed 24 July 2016]

more potentially interfering than the two CBOW models. With no clear correlation between the test scores evaluated in chapter 5 and our findings for the three selected models, it is difficult to determine which training parameters could generally excel another. Due to the scope of this thesis, it was not possible to make detailed observations of more than the three models. However, what became evident through this evaluation is that disambiguation and navigation downwards the hierarchical tree work remarkably well. We could observe that by not excluding members of other groups as the negative list for *3CosMul*, we still retrieve a majority of unseen words when conducting a new clustering. It prove a good decision to avoid predefining the number of clusters. When applying a constant number of 4 clusters, it was sometimes unclear how the bigger clusters could have formed. Nevertheless, we expect it will be necessary to restrict the number of different clusters in case of an exposition to users of an information retrieval platform. With more than a few colors, the word cloud quickly becomes confusing. We consider especially the experiments with iterative subclustering worth pursuing.

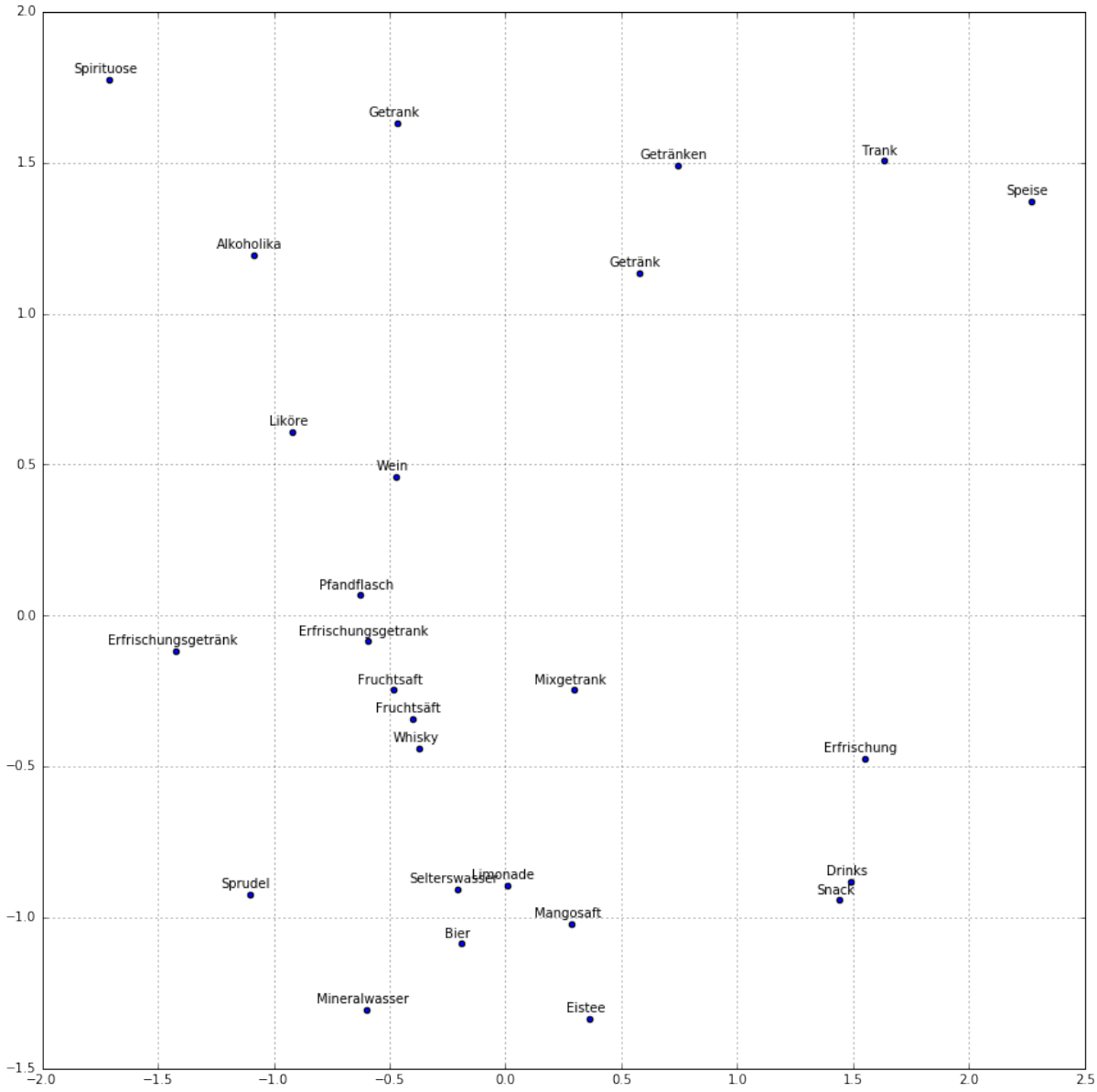


Figure 6.4: First principal components for *Getränk* (beverage). *Produced from* LemmaC *with randomised PCA*

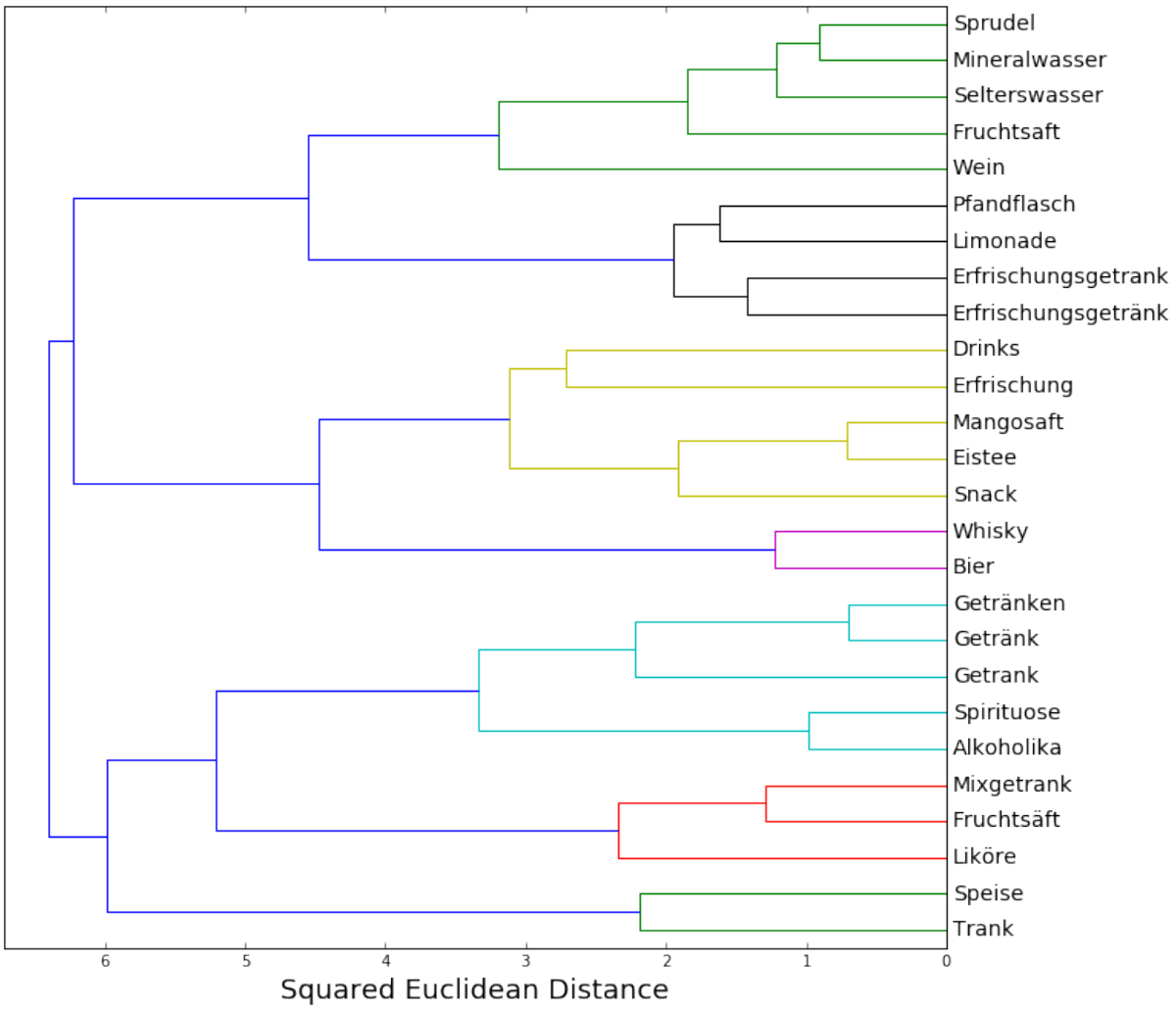


Figure 6.5: Clustering of *Getraenk* on 5 PCs. *Produced from* LemmaC, *corresponding to dendrogram in fig. 6.2c*

7 Conceptual Integration with an Enterprise Search Platform

Enterprise search solutions face several challenges: On the one hand, they need to handle company-specific requirements such as the indexation of various data sources or the compliance with multiple access levels. On the other hand, solutions have to address common problems of information retrieval, regarding e.g. ranking and presentation of results. This chapter offers a detailed analysis of how the language models and clustering techniques studied in previous chapters can be effectively deployed in enterprise search. Section 7.1 provides an overview of the components present in *intergator*, the platform under study. Ensuing, section 7.2 illustrates the points where word embeddings offer potential for improvement and where there arise opportunities for extended and innovative user support.

7.1 The intergator Search Platform

Primary value proposition of *intergator*, product of the company *interface projects GmbH*, is convenient access to internal company data. While selling also web search solutions, strong focus is on optimal support of enterprise search in all its specifics. The software is composed of multiple components, partially known also from other application areas of information retrieval (IR) such as generic web search, digital library or e-commerce search facilities. A screenshot of the search platform as deployed internally for development and test purposes is displayed in figure 7.1. The clean layout emphasises the different user interface components, but depends to a high degree on individual customer requirements.

Access to all file systems and data sources of the respective company is provided through individual connectors, via a single interface. Present and well established throughout IR applications is the search bar which invites users to enter their query formulated from an initial information need. The space below is reserved for the query results which can be presented as required in list (default), gallery or table format. The list view as depicted in figure 7.1 contains the following information:

- an icon or image preview for each result
- the document title
- the file path
- a text snippet, extracted from relevant information - with query terms emphasised in bold letters

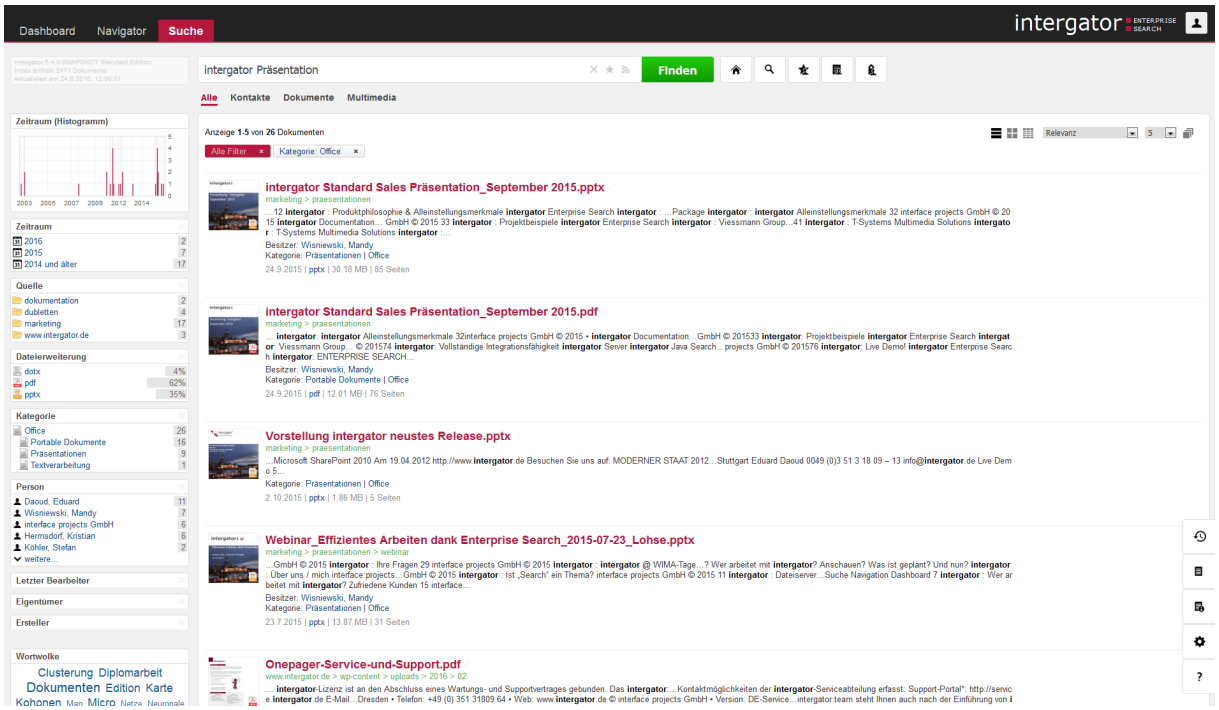


Figure 7.1: Intergator Search Interface. *The left column provides options to filter the results, e.g. for specific time ranges, data sources or file extensions. Applied filters are displayed as tags above the result list which enable the user to remove them or invert their effect.*

- the associated category
- the date of the last modification
- contacts associated with creation, modification or ownership of the document, senders and receivers in case of emails
- the size in terms of page numbers and/or byte size

All information is provided subject to applicability and availability. The left column contains filters which can be applied to restrict the results, e.g. to documents from certain time ranges, from a particular data source, or to retrieve only files with a specific extension. Applied filters are added as tags above the results and can be used as a control to invert or remove respective restrictions.

Separate to the classical search interface is the *Navigator*, displayed in figure 7.2, which enables an explorative search approach. Central part is a navigable tree as shown on the left side. Information to a document is displayed on selection from the list to the right in a respective detail view. Several charts can be configured to model details like the composition of results in terms of file extensions, the distribution over time or over people associated with the files. In figure 7.2 two representative analytics charts are depicted in the lower right corner. The bar chart visualises distribution of the result documents over categories such as *Office* or *Web*. The ring diagram displays the distribution over multiple document attributes, in the example: date, category, document type and file extension in respective order. Changes of the

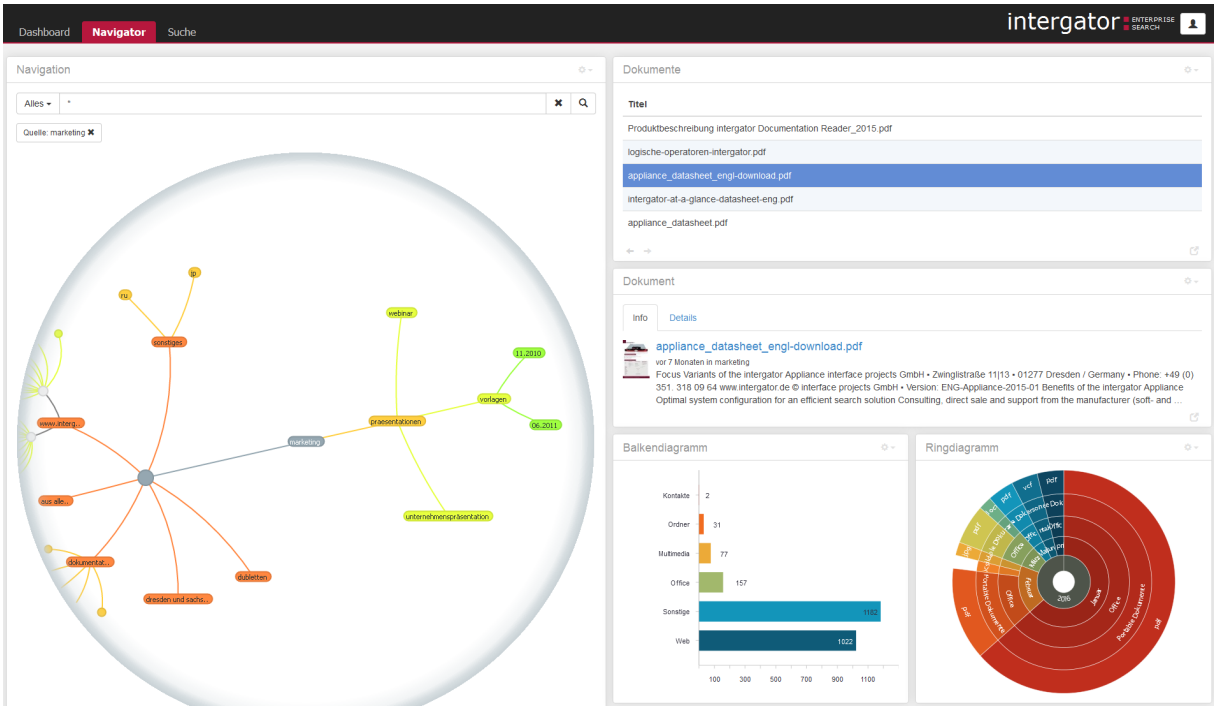


Figure 7.2: Intergator Navigator. *Explorative search controls give the user the option to navigate through the file tree while receiving visual feedback via analytics charts, in the example the distribution over categories, i.e. in combination with date, document type and file extension. Information for selected documents is displayed in a detail view above.*

result set induced by a new query or navigation in the search tree reflect in dynamic update of the analytics components. In general, all components can either be globally specified by administrators, offered to employees as fixed components or mere templates, or composition and configuration of components can be left entirely to the individual user.

The third building block of the intergator platform poses the social dashboard, the personal entry point for each user. An example of a possible configuration is depicted in figure 7.3. The composition includes a weather forecast, a list of popular documents, personally collected links and recently added contents, positioned in the left column. Organised in the middle is a search bar which, when used, redirects to the search page. The lower two widgets display results bookmarked by the user during earlier searches and the weekly menu of a canteen close to the office. Administrated help pages, corporate and public news feeds complement the dashboard with a third column. Popular supplements are an overview of upcoming dates synchronized with commonly used calendar software or arbitrary RSS feeds. Encouraging open exchange of information, page and single component configurations can be shared with other users, as well as conducted searches and bookmarked results.

Many of the functions presented here have been added over the last years. The volumes of data to be managed in individual customer installations become ever larger. This makes the results of the previously studied distributed language models relevant in two respects: On the one hand, there is constant interest in further ways to improve and support enterprise search,

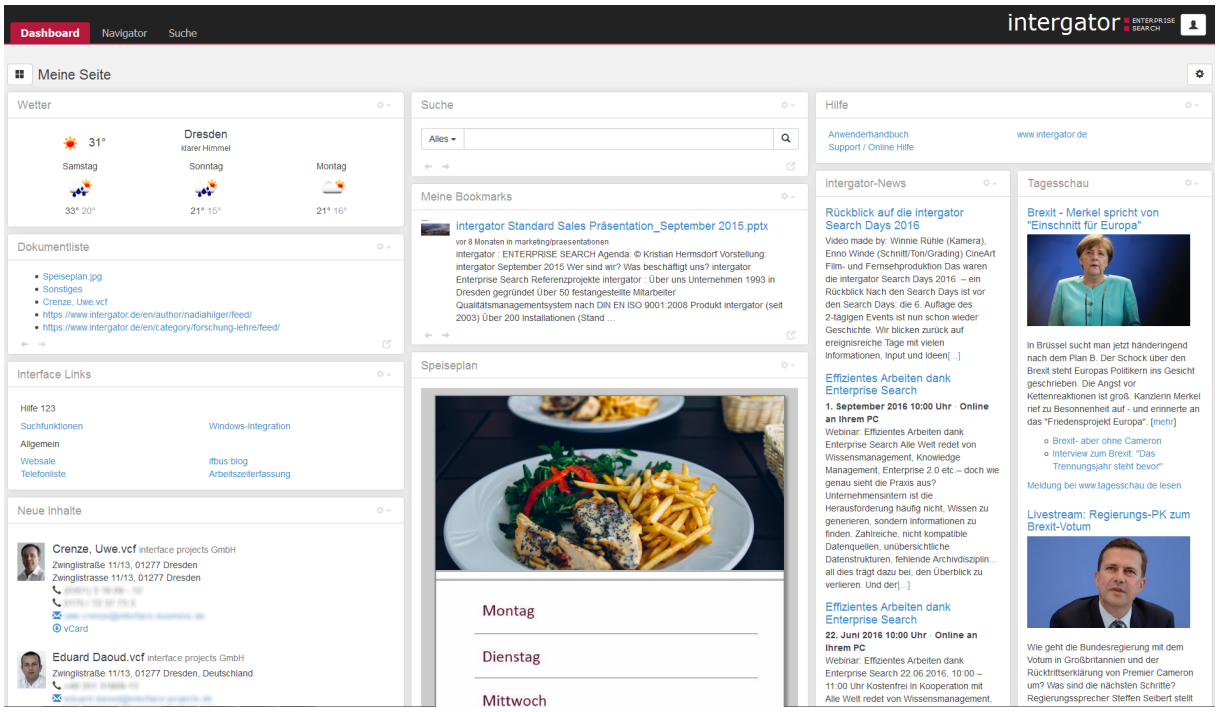


Figure 7.3: Intergator Dashboard. *The social dashboard can contain several components and constitutes the entry point for each user. Next to useful widgets such as weather forecast or corporate news ticker it can include content based on predefined queries and filters as well as documents bookmarked by the current or shared by other users.*

e.g. by means of more robust retrieval mechanisms or even completely new components. On the other hand, it requires performance gains compared to currently used technologies such as GATE[1] and Lucene[2] to be able to provide extended functionalities for respective amounts of data in the long run.

7.2 Deployment Concepts of Distributed Word Representations

The previous section provided insight into how employees of a company using *intergator* as their enterprise search solution are supported in their daily tasks. Subsequently, we present several deployment concepts of distributed word representations for improvement, but also for extension of the current search capabilities.

For illustration purposes we concentrate on the query term *books* in an imaginary data set of both German and English, content-wise corporate and more general documents. We assume a model with representations of both single tokens and phrases, to be recognised in a query by their unseparated occurrence and their existence in the model. Beside its obvious

[1] *General Architecture for Text Engineering*, see also `https://gate.ac.uk/` [accessed 12 July 2016]
[2] Java text search engine library, see also `https://lucene.apache.org/core/` [accessed 12 July 2016]

meaning as the English plural for a set of sheets fastened together between two covers, it also has a particular meaning within the company *interface projects*: Namely, it was the original designation for a component responsible for the preview of text documents with arbitrary file extension. While the official name has been changed to *Documentation Reader*, between developers, it is most commonly referred to by its original name. The sales department, however, has adapted the new name and uses it sometimes in full, sometimes in abbreviated form. As a result, a considerable number of terms are in circulation: *documentation reader, doc reader, reader, preview, books*, regardless of spelling mistakes or parsing errors in some places.

The example highlights some common difficulties that occur in enterprise search - different version names, company-specific vocabulary as well as an overlap of different word meanings. Furthermore, suppose the search user is looking for a specific piece of information which is not documented in digital form and would, in this case, like to consult an expert about the matter. Another information need arises which could be cured by a mature expert search. If a term had multiple meanings, it would be useful to be able to restrict the search to contents with the respective meaning. This equals a topic search and resembles available filter functionalities, which, however, can only be applied to the confirmed query. Practice and development trends in web search show that this support does not always suffice. In some cases, the steps towards the desired result are too many. Wide distribution of search hits in a variety of different categories complicates the right choice, the user either aborts the search or expends considerable time going through an unnecessarily large result set.

7.2.1 Improved Document Retrieval

The primary requirement is to be able to retrieve all relevant results for a given query, be it the one document including a misspelled or incorrectly parsed occurrence of the term, the sales presentation or the open development ticket. For default search behaviour we therefore suggest an automatic query expansion without directly involving the user.

In multidimensional feature space of distributed word representations, the maximum cosine-based similarity is 1 (between identical words). The distance to the top most similar words depends on the word's ambiguity itself and the terminological diversity. If a word has multiple meanings as *books* in the company-specific context of *interface projects*, it is not likely to find an almost identical term. However, due to a very strong representation of the term in the corporate-specific meaning, the most similar terms by far are related to the concept of the thereby designated software component. If a word has no close substitutes in any context represented in the training corpus, it is accordingly difficult to find similar vocabulary entries.

As illustrated by the given example, it could be a mistake to expand the query by an absolute number of most similar terms. We suggest to consider both, a minimal cosine similarity and a maximum number of additional terms to keep the additional computational effort manageable.

7.2.2 Improved Query Suggestions

Above, we referred to the difficulty of a precise query formulation, leaving the user easily frustrated - a lack of usability is perceived. However, for document-centered information

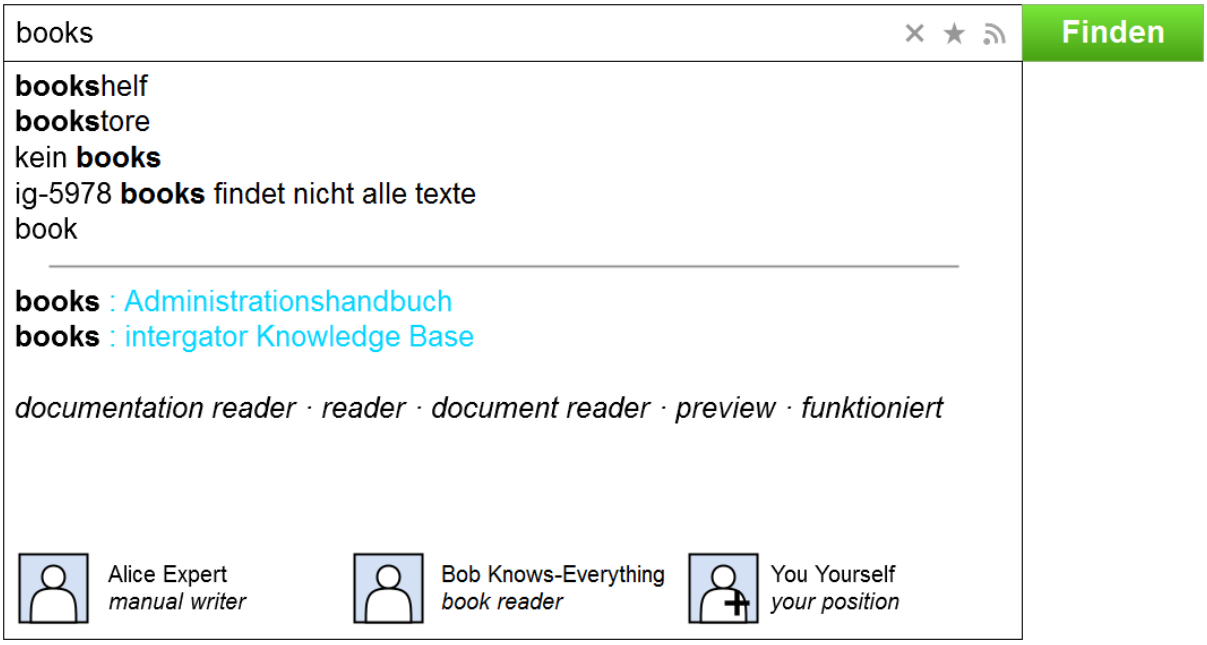
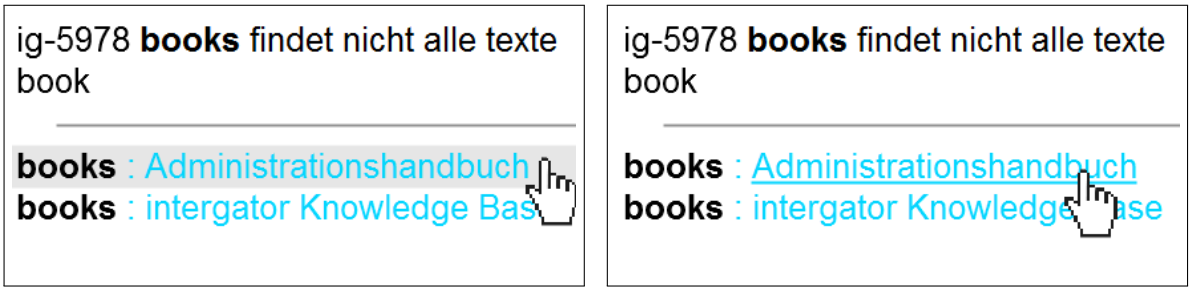


Figure 7.4: Expanded Search Bar. *On entering a query, the user is provided with assistance of two kinds: At the top he finds various auto-suggestions for textual completion or correction. The different elements below enable additional navigation and ad-hoc search refinement as applicable for the exact input.*

needs, the search bar in many cases poses a fast and solid solution. What alternatives are there to lead the user to an improved result set at the stage when entering the query? An example how viable search refinements can already be proposed during typing is depicted in figure 7.4. The first five entries are suggestions to auto-complete or correct the given query based on search logs (frequency and timeliness of similar queries). This functionality is currently supported and can accelerate search and thinking process, including consideration of possible typing mistakes.

Suggestions below the midline relate to the exact query. The first two entries enable direct refinement to the two most probable subject areas. According to the currently implemented workflow, the individual occurrences in documents classified as such have to be considered. Supported by distributed word representations and subject area representations averaged over contained words, a semantic similarity can be calculated which does not have to change with a single document update but is more stable and, at the same time, less costly to query. In addition, based on the system's representation of semantic similarity, direct navigation into topic areas can be provided. Beside confirming the refinement to a subject area (figure 7.5a), users can click and follow a link to the area (figure 7.5b) simply by entering associated terms, i.e. without having to know about the complete contents and the explicit title.

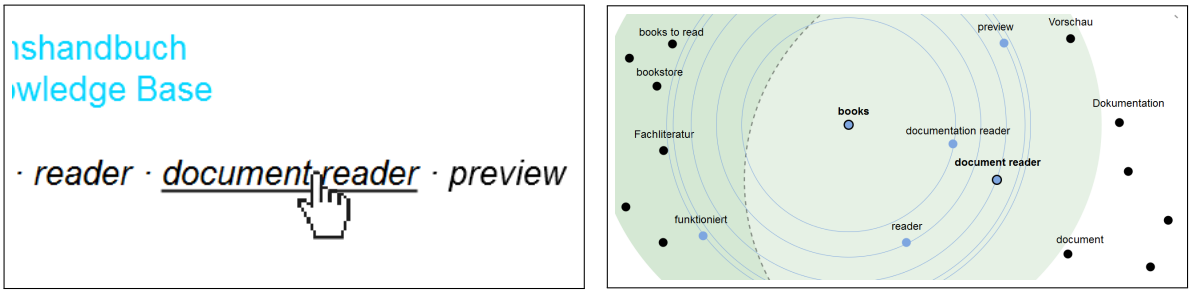
The entries below written in italic letters constitute terms with the highest semantic similarity towards the query. Results from chapter 5 prove that, while outliers can occasionally occur, cosine similarity between distributed word representations is a very good indicator for



(a) Topic-based search refinement (b) Link to topic-associated contents

Figure 7.5: Topic Area Suggestions. *Based on semantic similarity to the given query, the user can refine his search via the expanded search bar to an associated topic area or follow the link to an overview of its contents.*

semantic similarity. Clicking on one of the terms (see also figure 7.6) executes the respective search instead of the original query. It is important to note that this can - despite automatic query expansion - lead to different results. Although the example query *books* might be automatically expanded by the most similar words as suggested in section 7.2.1, this can cause a query shift, highlighted in figure 7.6b: For the query *books*, the most similar terms were *documentation reader*, *reader*, *document reader*, *preview* and *funktioniert* (functions). Example additional terms in close proximity to *books* are *Fachliteratur* (technical literature), *books to read* and *bookstore*. If the query were to be automatically expanded by these terms, results returned would clearly differ from those of the selected search *document reader*. Although a shared expansion by *preview*, *documentation reader* and *reader*, apart from the original term *books*, the query could also be expanded by terms such as *document* and *Dokumentation* (documentation).

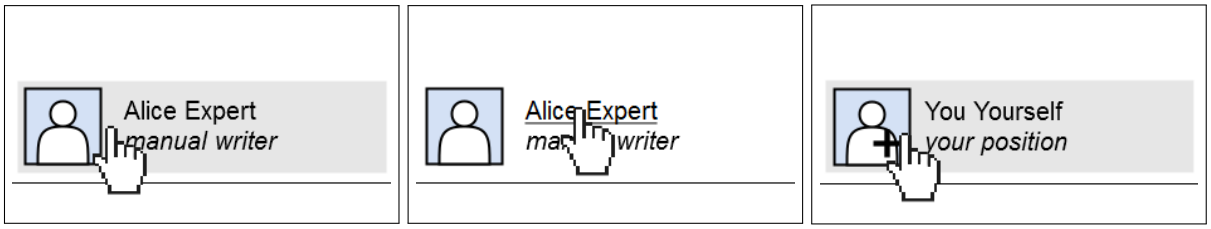


(a) Link to related search (b) Query shift with related search

Figure 7.6: Related Term Suggestions. *Based on semantic similarity to the given query, the user can shift or refine his query by following the link to search for a related term. Results will not be distinct but may better define the underlying information need.*

The last row displays contacts suggested as confirmed experts to the query. By selecting the whole item as depicted in figure 7.7a, the user can refine his search to documents associated with the contact as being their author, modifier or owner. If the user wants to know whom to consult directly about a given matter, he can find the person by clicking on the contact

name (fig. 7.7b). Theoretically, support of automatic expert finding is possible without manual confirmation. However, to protect privacy rights, it is important to leave the final decision whether or not to confirm one's expertise to the respective user. A conceivable solution would be to let users also recommend arbitrary contacts as experts, also left to be confirmed. Similar mechanisms are built into social platforms such as *facebook*, where depending on respective commonalities, suggestions to add this commonality to a friend's profile are proposed. The alternative presented in figure 7.7c is to support only adding oneself. This seems reasonable from the perspective that, even if the current user is entering searching for documents about given matter at the moment, in many cases, this is due to his dealing with the matter professionally. Making the confirmation as easy as possible via one-click confirmation encourages the otherwise unlikely participation.



(a) Expert-based search refinement

(b) Link to expert-associated contact information and contents

(c) Confirmation of expertise on the given matter

Figure 7.7: Expert Suggestions. *Based on semantic similarity to the given query, the user can refine his search via the expanded search bar to contents associated with an acknowledged expert or follow the link to an overview of respective contact information and associated contents. Additionally, the user can - through one click - confirm his personal knowledge about the given matter.*

7.2.3 Additional Support in Explorative Search

In section 7.1 the platform's explorative search component was presented, which essentially enables a search through the filing structure of all integrated storage systems. While this can be of great value for very strictly organised content and users who work with the file system, it does consider the actual content only secondary, i.e. the filter to documents at a (single) selected location is applied before computing the query term. Nevertheless, explorative search constitutes a popular and promising approach to facilitate efficient information retrieval. Several research and techniques actually used are discussed by Hearst [17]. Motivated by the evaluation of clustering capabilities in chapter 6, figure 7.8 depicts a possible application to explorative search. The example shows a navigable wordcloud for the search term *Getränk*[3] (beverage). By clicking on one of the related-term clusters, each represented by a different font color, the user can refine his search to index objects additionally containing one or more terms of the given cluster. With each iterative refinement, cluster representations are replaced by

[3] we refrain from using the example *books* in return for a valid example produced with our selected lemmatised CBOW model

(a) Initial query *Getränk* (beverage)

(b) Refinement and subclustering of the green cluster (mainly refreshments)

Figure 7.8: Navigable Word Cloud. *As additional explorative search component, the user can navigate through a semantic word cloud. Assumed here is a navigation from general beverages to refreshments.*

their containing subclusters, enabling a finer distinction, and accordingly, selection of a result subset. The advantage with a UI component like this as opposed to concrete disambiguation suggestions such as "Did you mean...?"[4] is that it is more robust to potential outliers. Closely related terms can occasionally fall into different subclusters, e.g. *Erfrischungsgetränk* (refreshment drink) and *Erfrischung* (refreshment) belong to two separate clusters, but may occur again in each of the subclusters. The navigable word cloud implicitly serves for general search refinement and word sense disambiguation alike. Provided a set of terms as definition of each semantic content group, the user quickly gains an overview and can make the right choices.

[4]Disambiguation suggestions requiring determination of the abstract concept formed part of the broader vision for this work

8 Conclusion

In the course of this thesis, a number of subtasks were tackled with the overall goal of providing improved and extended search capabilities with good performance and adaptability towards different business customers. Section 8.1 summarises the contents and results of previous chapters. We conclude this work with a discussion of associated scientific problems and opportunities for future research.

8.1 Summary

Initially, an overview of important literature was presented in chapter 2. Besides foundational research on modelling natural language as patterns of co-occurrence, we pointed out relevant articles on the utilisation of neural networks in machine learning. We completed the overview with literature on state-of-the-art word embedding techniques, the evaluation of resulting models and publications on cluster analysis.

In chapter 3, we discussed the foundations and general capabilities of distributed word representations. A look into the history of natural language modelling, specifically that based on the co-occurrence of words, revealed that research has long been engaging in the development of individual algorithms which can be identified in modern implementations. The beginnings reach back as far as to the 1950s, when computers were not even a part of everyday life.

Advances in hardware and software, i.e. the facilitation of parallelisation and, first and foremost, improved computing and memory capacities, enable the application of accumulated knowledge and research to today's problems. A promising combination of different algorithms and concepts poses the toolkit word2vec, whose functionalities were analysed in detail in section 3.5. Among other language modelling techniques that are currently being researched, word2vec has gained the focus of attention due to a particularly efficient implementation. Whereas many statistical approaches tend to concentrate on exact calculations of probabilities, word2vec mimics the functionalities of biological neurons and strives to attain the best possible solution using local optimisation and approximation. With this approach, it directly makes use of the error robustness and excellent feature projection capabilities of artificial neural networks.

For an effective contribution to enterprise search, further attention was directed to clustering of distributed word representations and its potential use. On the one hand, cluster analysis is an established means for the qualitative assessment of data. On the other hand, the results offer specific opportunities for visualisation, which increasingly support users in information retrieval tasks. This trend can also be observed in common fields such as general web search or e-commerce. A profound decision for any clustering algorithm required a sys-

tematic overview of techniques. This overview was provided in chapter 4. Apart from classical procedures like hierarchical clustering or k-means, we also discussed possibilities to deal with high dimensionality.

In chapter 5, we conducted an extensive evaluation of embedding models, essential basis for a sound evaluation of clusterings. Although we restricted ourselves to the toolkit word2vec, a single corpus and a subset of parameters, several factors had to be evaluated. The eventual analysis followed trainings and a computation of test scores for over 200 models.

Within the course of this thesis, we could not examine each and every mutual impact between the different training parameters to a full extent. Nevertheless, this thesis presents, to the best of our knowledge, the most extensive examination of distributed word representations based on word2vec for the German language. Even considering reports on the morphologically less diverse language English, there are few publications that offer statistically relevant and unambiguous insight. In this context, we want to stress the fact that many evaluation possibilities are criticised for their lack of (universal) significance. Respective drawbacks in combination with parameter influences were discussed after the training series in section 5.4. A number of promising models were selected to be used in subsequent clustering experiments Due to the extensive experiments, the training and evaluation of lemmatised models including phrases in combination with promising parameters[1], could not be conducted in this work.

Only during training and evaluation, started early and continued for the greater part of this assignment, several challenges became evident to which we had to devote the necessary time. For example, we had not expected the degree of complexity in arriving at a valid assessment, even with or especially due to the sheer number of collected test results. Although our interactive working environment prove an otherwise valuable asset, only transferring the model evaluation results to Excel allowed us to efficiently work with the data and obtain an overview crucial for the task. In general we observed a low overall correlation between different test sets which made the selection of models for our following cluster experiments very difficult. In the end, we decided for a conservative approach and filtered only models with values below the average from the list. Also, we had to confirm the low results for the test set of paradigmatic semantic relations.

In chapter 6 we conducted clusterings for our final selection of distributed word representations which we had initially tested with early models. A qualitative assessment revealed a general advantage of CBOW over skip-gram which we could not derive directly from the test scores. In compliance with the poor performance on hyponym analogy tasks, it was not possible to automatically derive the projection from a single word to the overarching concept. With this result, we confirm the findings of publications such as [22] that paradigmatic relations such as hyponymy or antonymy cannot yet be extracted from word embedding models with current approaches. Interesting in this regard would be whether there are other methods better suited to finding such relations than cosine similarity between normalised vectors or *3CosMul* as the multiplicative combination for completion of analogy tasks.

In parallel to the protracted studies, we experimented with clustering techniques and visualisation options which could create further value for enterprise search users - besides discovering results based on semantic resemblance. To assess the potential of word embeddings in depth and provide the reader with a concrete understanding of the system, we began chapter 7 with

[1] for our purposes, determined in the extrinsic evaluation

an introduction of indiviual components of the enterprise knowledge management platform *intergator*.

Accordingly, the first section demonstrated the wide range of supporting functionalities available to platform users. Feedback collected from various customer projects offers two major insights: On the one hand, filter options for subsequent query refinement are rarely exhausted. This can most likely be attributed to the user's perception that the selection of the appropriate filter is too complicated, either due to a disparity between the search process and the user's train of thought, or due to the additionally required UI interaction - each filter option is uniformly presented, regardless which is semantically most probable considering the query. On the other hand, there is a strong interest in further intelligent search support.

As a conceptual improvement based on these insights, we proposed an expanded search bar which offers direct query refinement alongside the expected auto-suggestions for input correction and completion. On the basis of semantic word vectors from distributed language models, semantic vectors can also be assigned to relevant facets, categories or contents at important file locations. Similarly, this can be applied to users: Even though a sensitive privacy policy requires confirmation from the respective person, based on their digital contributions, it is possible to suggest a probable expert which can be consulted on a specific question or topic. Since the unsupervised approach of clustering could not supply reliable results in terms of automatic hypernym finding, clustering cannot offer a direct advantage to these tasks. Implicitly, clustering can help in the general assessment of models or with a specific analysis of unexpectedly low performance in the individual case.

However, the situation is different for explorative search components. Finally, we proposed word cloud navigation as promising concept of a search platform extension. In an example, we illustrated that hierarchical clustering algorithms directly support semantic navigation, relieving us from the need to name a concrete hypernym. The interactive component enables seamless disambiguation and refinement with the advantage of being applicable to any query, however fine-grained or abstract, whether composed of one or multiple search terms.

8.2 Further Work

This thesis contributes valuable insights for both scientific research and the company *interface projects GmbH*, which has an interest in applying distributed language models for improvement and extension of its enterprise knowledge management platform *intergator*. Nevertheless, there is a significant number of open questions which were revealed in the course of this work and pose an entry point for further research. The extensive evaluation of neural embedding models provides a statistical basis for additional tuning of configuration parameters to a respective task. Alongside this thesis, a wide range of experiments were conducted. As an example, a successful use case demonstrated that, linking two models by a selection of shared concepts, the search space can be extended to additional topic areas and languages. This is especially interesting for enterprise search, where semantic search could largely benefit from a combination of the individual corporate concepts with common 'world' concepts. Following the example of the experimental setup described in chapter 5, we can now better assess and compare the models for such tasks. However, we could also demonstrate by a qualitative analysis in chapter 6 that

the test scores do not suffice for a final assessment. Further test series may show whether, for a corpus with numerous multi-word units and a beneficial configuration, the model quality can actually improve. With the word cloud navigation concept proposed in chapter 7, we presented a valuable extension to enterprise search which we would be interested to see put to actual use.

Bibliography

[1] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics, 2009.

[2] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. RAND-WALK: A latent variable model approach to word embeddings: Version 6 (2016). *CoRR*, abs/1502.03520v6, 2016.

[3] K. Backhaus, B. Erichson, W. Plinke, and R. Weiber. *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. Springer-Verlag, 12th edition, 2008.

[4] M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247, 2014.

[5] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[6] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Z. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng. Large Scale Distributed Deep Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pages 1223–1231. Curran Associates, Inc, 2012.

[7] J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[8] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

[9] M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer. Problems With Evaluation of Word Embeddings Using Word Similarity Tasks. *arXiv preprint arXiv:1605.02276*, 2016.

[10] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing Search in Context: The Concept Revisited. In *Proceedings of the 10th International World Wide Web Conference (WWW 2001)*, pages 406–414, 2001.

[11] J. R. Firth. A synopsis of linguistic theory 1930-55: Selected papers of J.R. Firth 1952-59. *Reprinted in: Palmer, F. R. (ed.) (1968). Selected Papers of J. R. Firth 1952-59, pages 168-205. Longmans, London.*, pages 1–32, 1957.

[12] Y. Goldberg and O. Levy. Word2Vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method. *arXiv preprint arXiv:1402.3722*, 2014.

[13] I. Gurevych. Using the structure of a conceptual network in computing semantic relatedness. In *Natural Language Processing (IJCNLP 2005)*, pages 767–778. Springer, 2005.

[14] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, pages 297–304, 2010.

[15] W. K. Härdle and L. Simar. *Applied Multivariate Statistical Analysis*. Springer Science & Business Media, 4th edition, 2014.

[16] Z. S. Harris. Distributional Structure. *Word*, 1954.

[17] M. A. Hearst. *Search User Interfaces*. Cambridge University Press, Cambridge, England, 2009. `http://searchuserinterfaces.com/book/` [accessed 10 May 2016].

[18] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Psychology Press, 2005.

[19] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Foundations*, volume 1, pages 77–109. MIT Press, Cambridge, Massachusetts, 1989.

[20] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science and Engineering*, 9(3):90–95, 2007.

[21] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 4 July 2016].

[22] M. Köper, C. Scheible, and S. Schulte im Walde. Multilingual reliability and "semantic" structure of continuous word spaces. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS 2015)*, pages 40–45, 2015.

[23] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1, 2009.

[24] T. K. Landauer and S. T. Dumais. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104(2):211–240, 1997.

[25] O. Levy and Y. Goldberg. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308, 2014.

[26] O. Levy and Y. Goldberg. Linguistic Regularities in Sparse and Explicit Word Representations. In *Proceedings of the 18th Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan, 2014. Association for Computational Linguistics.

[27] O. Levy and Y. Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 2177–2185, 2014.

[28] O. Levy, Y. Goldberg, and I. Dagan. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[29] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, Massachusetts, 1999.

[30] W. McKinney. Data Structures for Statistical Computing in Python. In S. van der Walt and J. Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51–56, 2010.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013.

[32] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119, 2013.

[33] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic Regularities in Continuous Space Word Representations. In *2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)*, pages 746–751, 2013.

[34] M. L. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, expanded ed., 4th print. edition, 1990.

[35] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 1081–1088, 2009.

[36] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 2265–2273, 2013.

[37] A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

[38] S. Mohammad, I. Gurevych, G. Hirst, and T. Zesch. Cross-Lingual Distributional Profiles of Concepts for Measuring Semantic Distance. In *EMNLP-CoNLL*, pages 571–580, 2007.

[39] F. Morin and Y. Bengio. Hierarchical Probabilistic Neural Network Language Model. In R. G. Cowell and Z. Ghahramani, editors, *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics, 2005.

[40] L. Parsons, E. Haque, and H. Liu. Subspace Clustering for High Dimensional Data: A Review. *ACM SIGKDD Explorations Newsletter*, 6(1):90–105, 2004.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[42] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empiricial Methods in Natural Language Processing (EMNLP 2014)*, volume 12, pages 1532–1543, 2014.

[43] F. Pérez and B. E. Granger. IPython: a System for Interactive Scientific Computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007.

[44] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[45] M. Rei and T. Briscoe. Looking for Hyponyms in Vector Space. In *Proceedings of the 18th Conference on Computational Natural Language Learning*, pages 68–77, Ann Arbor, Michigan, 2014. Association for Computational Linguistics.

[46] F. Rosenblatt. *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, 1962.

[47] H. Rubenstein and J. B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

[48] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *NATURE*, 323(6088):533–536, 1986.

[49] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Internal Representations by Error Propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Foundations*, volume 1, pages 318–362. MIT Press, Cambridge, Massachusetts, 1989.

[50] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Psychological and Biological Models*, volume 2. MIT Press, Cambridge, Massachusetts, 8th edition, 1988.

[51] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, volume 1. MIT Press, Cambridge, Massachusetts, 9th edition, 1989.

[52] S. Scheible and S. Schulte im Walde. A database of paradigmatic semantic relation pairs for German nouns, verbs, and adjectives. In *Proceedings of the COLING 2014 Workshop on Lexical and Grammatical Resources for Language Processing*, pages 111–119, 2014.

[53] S. Schmidt, P. Scholl, C. Rensing, and R. Steinmetz. Cross-lingual recommendations in a resource-based learning scenario. In *Towards Ubiquitous Learning*, pages 356–369. Springer, 2011.

[54] T. Schnabel, I. Labutov, D. Mimno, and T. Joachims. Evaluation methods for unsupervised word embeddings. In L. Màrquez, C. Callison-Burch, and J. Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 298–307, Lisbon, Portugal, 2015. Association for Computational Linguistics.

[55] StatSoft Inc. *Electronic Statistics Textbook*. StatSoft, Tulsa, Oklahoma, 2013. `http://www.statsoft.com/textbook/` [accessed 15 July 2016].

[56] P. D. Turney and P. Pantel. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, 2010.

[57] S. v. d. Walt, S. C. Colbert, and G. Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science and Engineering*, 13(2):22–30, 2011.

[58] B. J. Wilson and A. M. J. Schakel. Controlled Experiments for Word Embeddings. *arXiv preprint arXiv:1510.02675*, 2015.

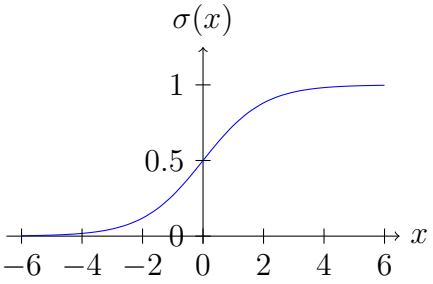
[59] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

List of Figures

List of Tables

Appendix

A Logistic Function



The *logistic function* or *sigmoid function* is, due to its simple differentiability, a common choice of activation function for artificial neurons. It is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

with the characteristics of being

bounded,
: *All values of $\sigma(x)$ lie between 0 and 1.*

differentiable and
: *A derivative exists at each point in its domain. The graph has no breaks, bends, nor cusps.*

real-valued.
: *The function is defined for $x \in \mathbb{R}$.*

Step-wise derivation:

$$\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1 + e^{-x}} \\
&= \frac{d}{dx}(1 + e^{-x})^{-1} \\
&= -(1 + e^{-x})^{-2} \cdot -e^{-x} \\
&= \frac{e^{-x}}{(1 + e^{-x})^2} \\
&= \frac{1}{1 + e^{-x}} \cdot \frac{e^{-x}}{1 + e^{-x}} \\
&= \frac{1}{1 + e^{-x}} \cdot \frac{1 + e^{-x} - 1}{1 + e^{-x}} \\
&= \frac{1}{1 + e^{-x}} \cdot (1 - \frac{1}{1 + e^{-x}}) \\
&= \sigma(x) \cdot (1 - \sigma(x))
\end{aligned}$$

B Model Evaluations

The following pages provide an overview of model scores per parameter variation and test set. For analysis and comparison to benchmarks reported in other publications, please refer to section 5.4 on page 45.

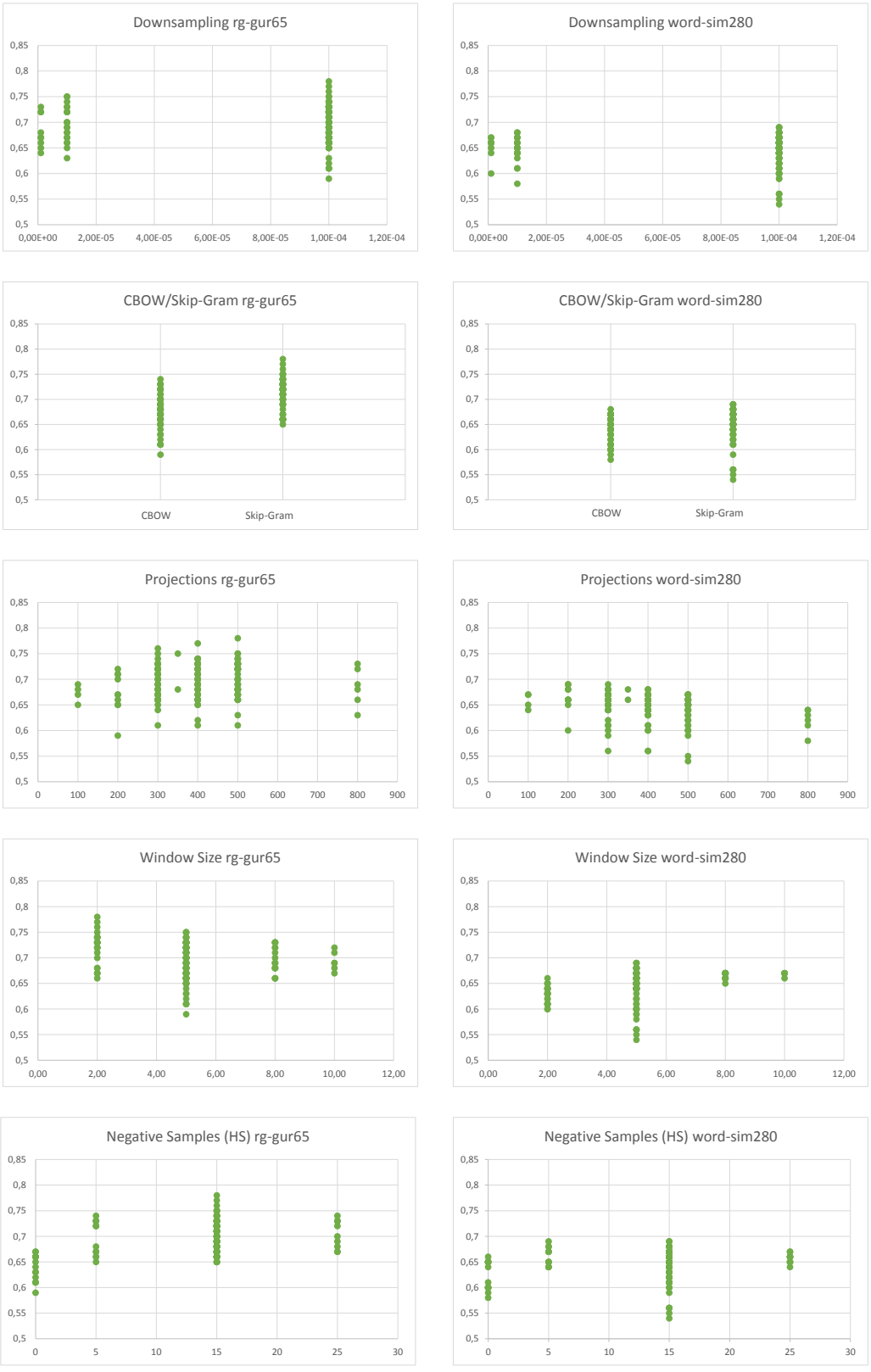


Figure 8.1: Spearman's Rank Correlation for Test Set *rg-gur65* and *word-sim280*. *Spearman's ρ models the non-linear correlation between human-assigned similarity scores and cosine similarities over each 65/280 related and synonym word pairs.*

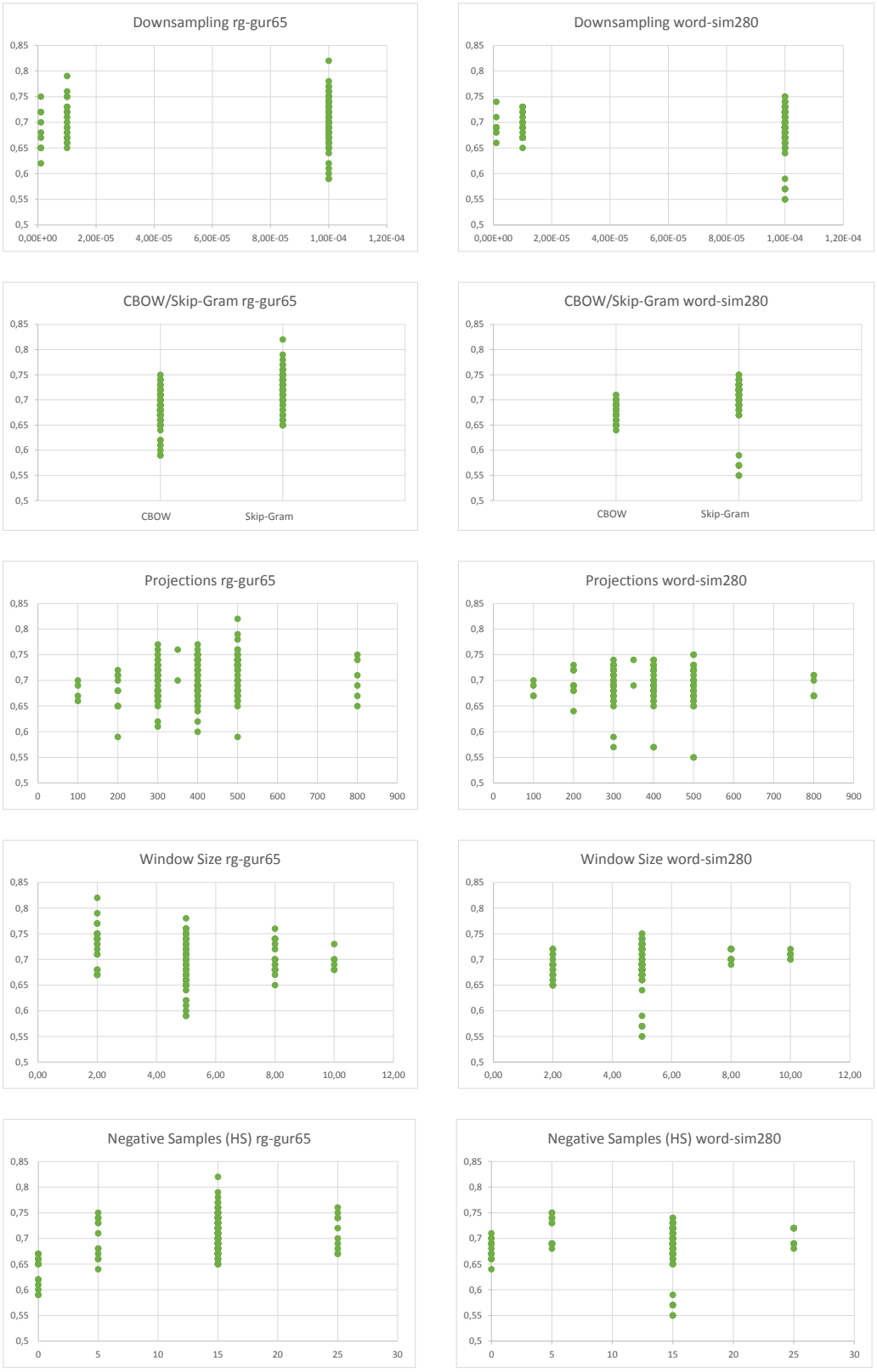


Figure 8.2: Pearson Correlation for Test Set *rg-gur65* and *word-sim280*. *Pearson's r models the linear correlation between human-assigned similarity scores and cosine similarities over each 65/280 related and synonym word pairs.*

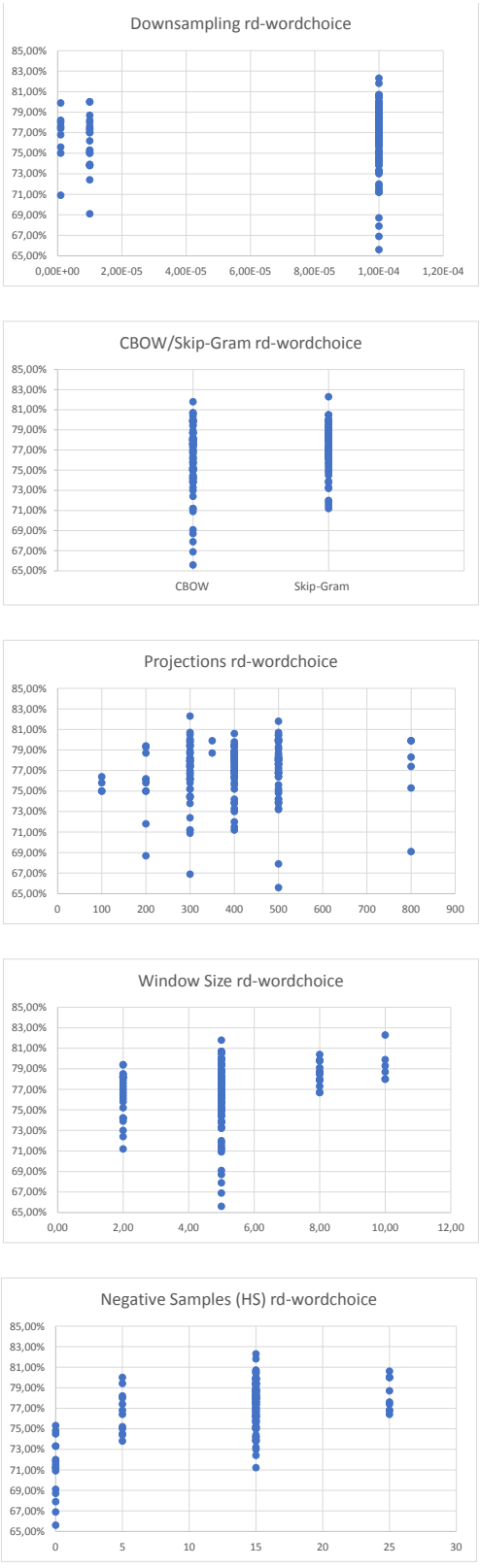


Figure 8.3: Accuracy for Closed Synonym Questions. *Each model is represented with a score of how many times overall the synonym was identified correctly among 4 possible choices. The test collection created from issues of the German Reader's Digest contains 426 multiple choice questions.*

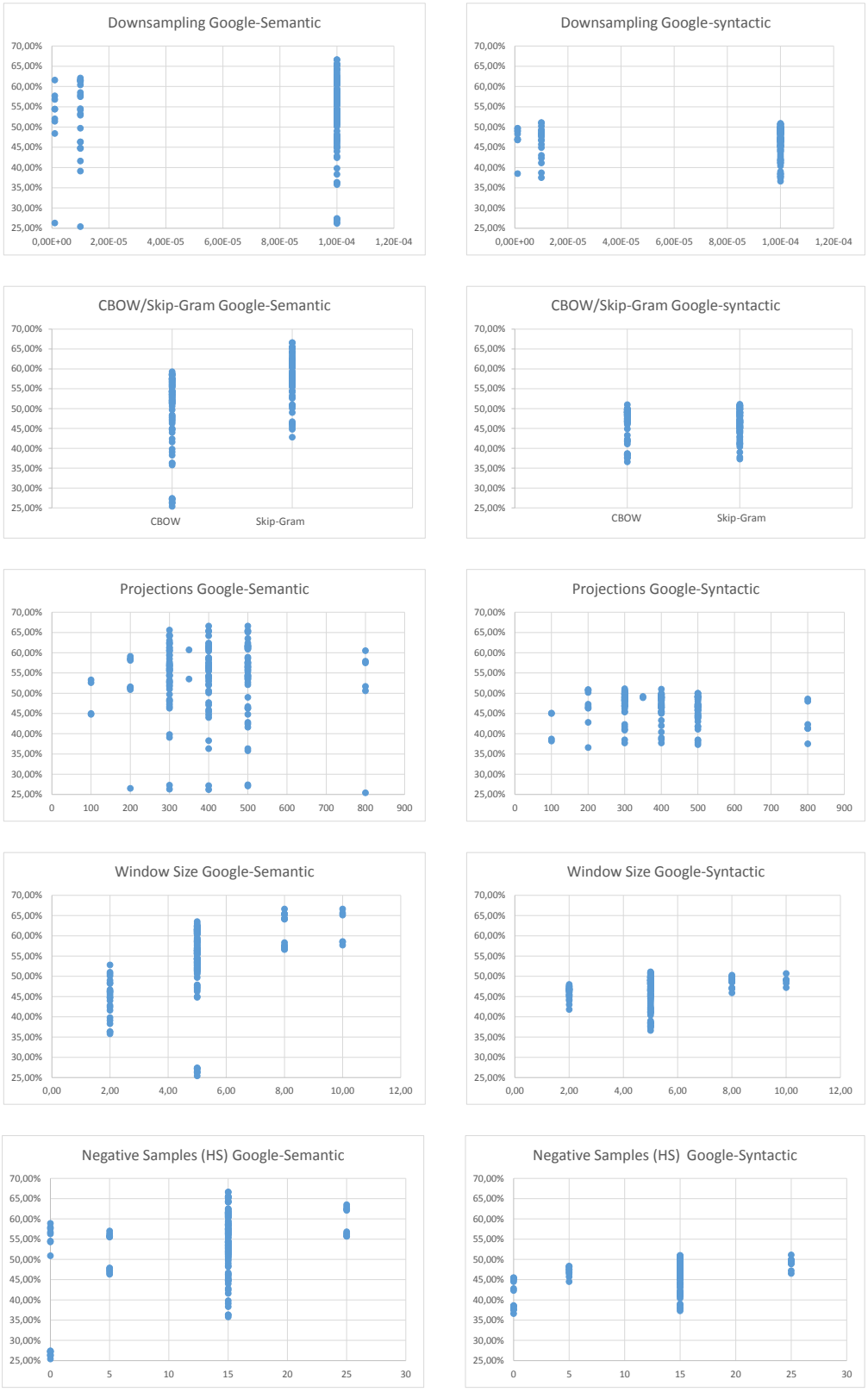


Figure 8.4: Accuracy for Google Analogy Test Set. *Each model is represented with a score of how many times overall the second analogy word pair was complemented correctly, computed with 3CosMul. Scores are presented separately for semantic and syntactic task sections.*

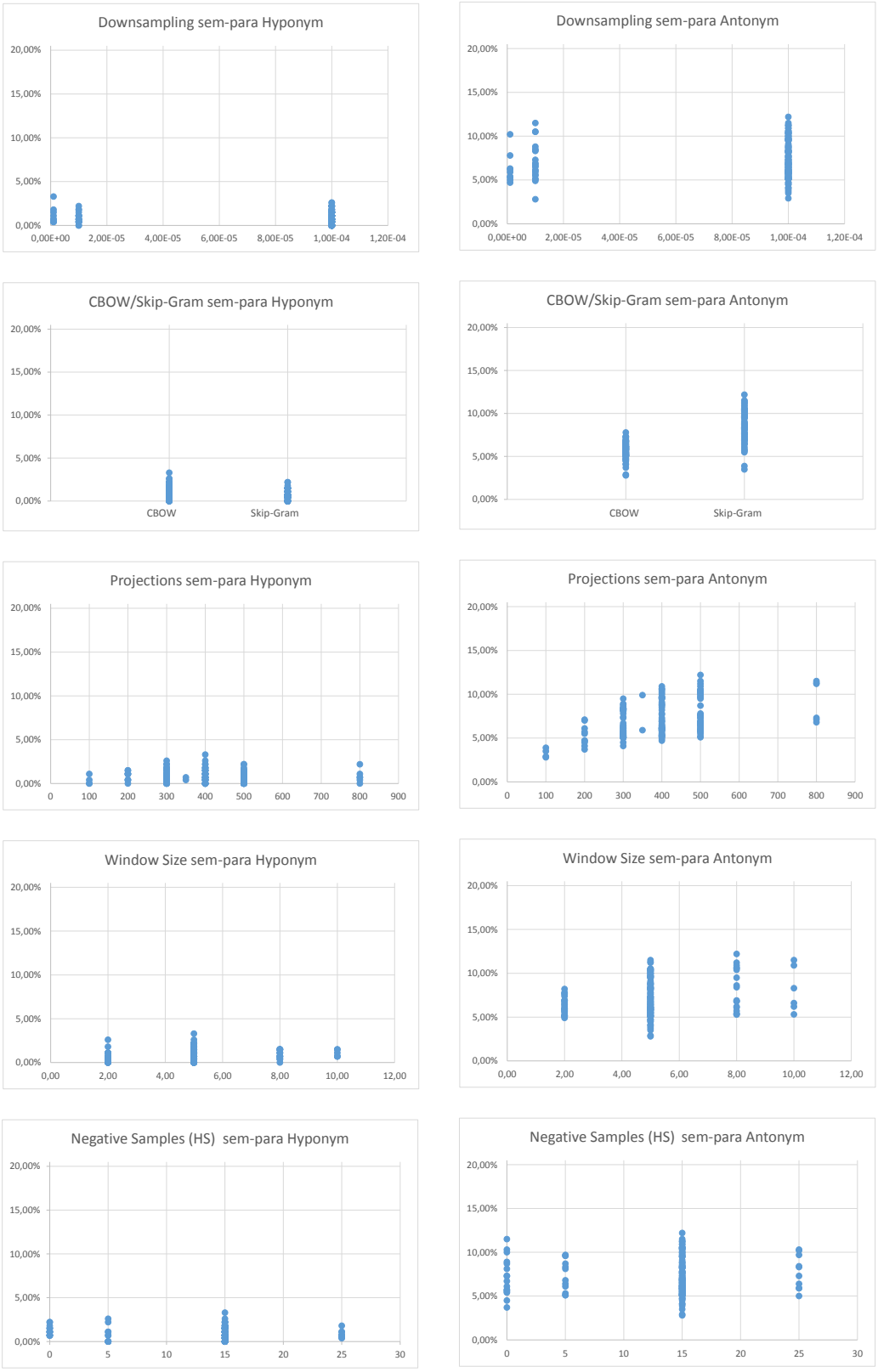


Figure 8.5: Accuracy for *sem-para* Test Set (1). *Each model is represented with a score of how many times overall the second analogy word pair was complemented correctly, computed with 3CosMul. Scores are presented separately for hyponym, synonym, antonym task sections and in total.*

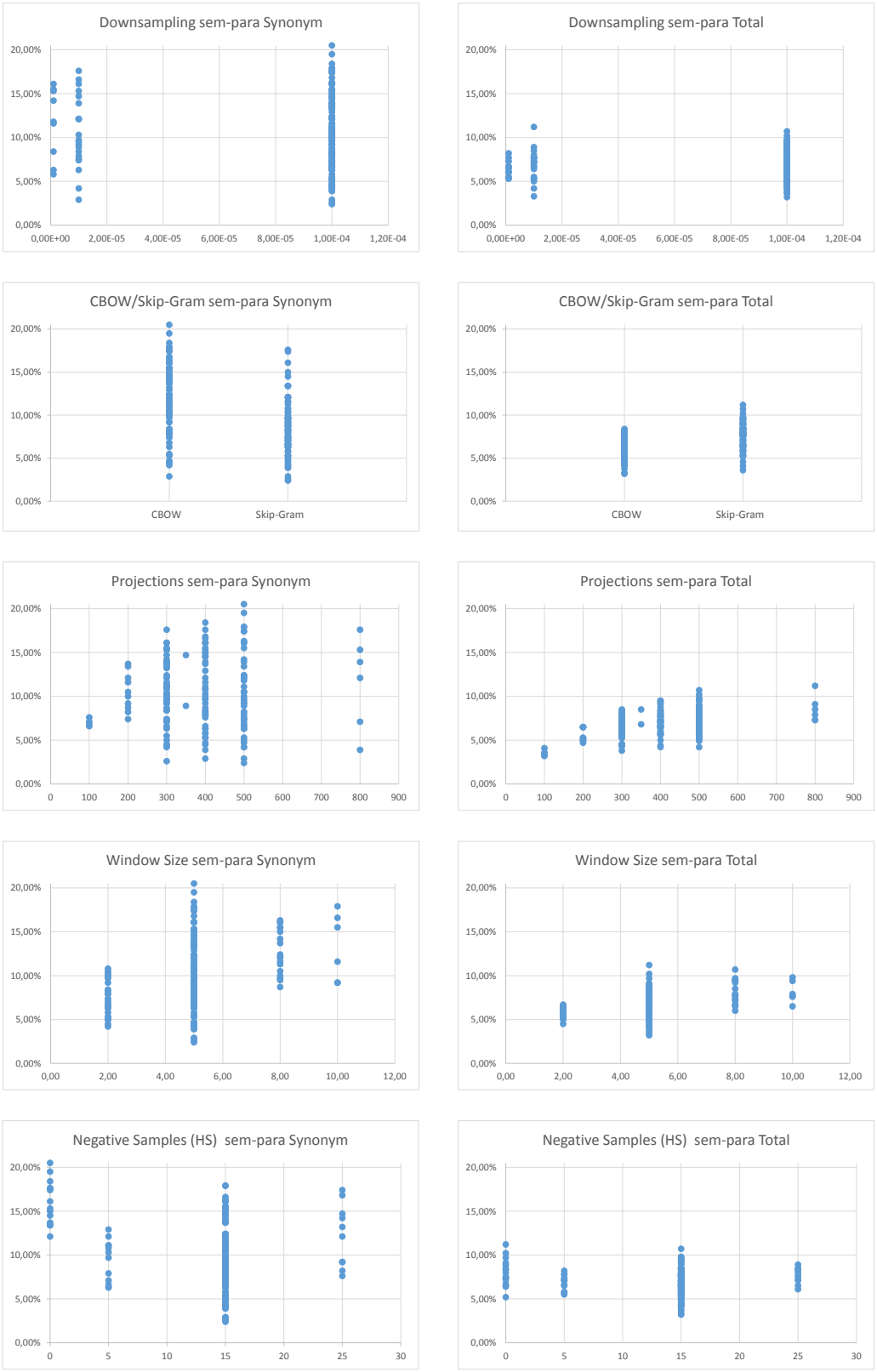


Figure 8.6: Accuracy for *sem-para* Test Set (2). *Each model is represented with a score of how many times overall the second analogy word pair was complemented correctly, computed with 3CosMul. Scores are presented separately for hyponym, synonym, antonym task sections and in total.*

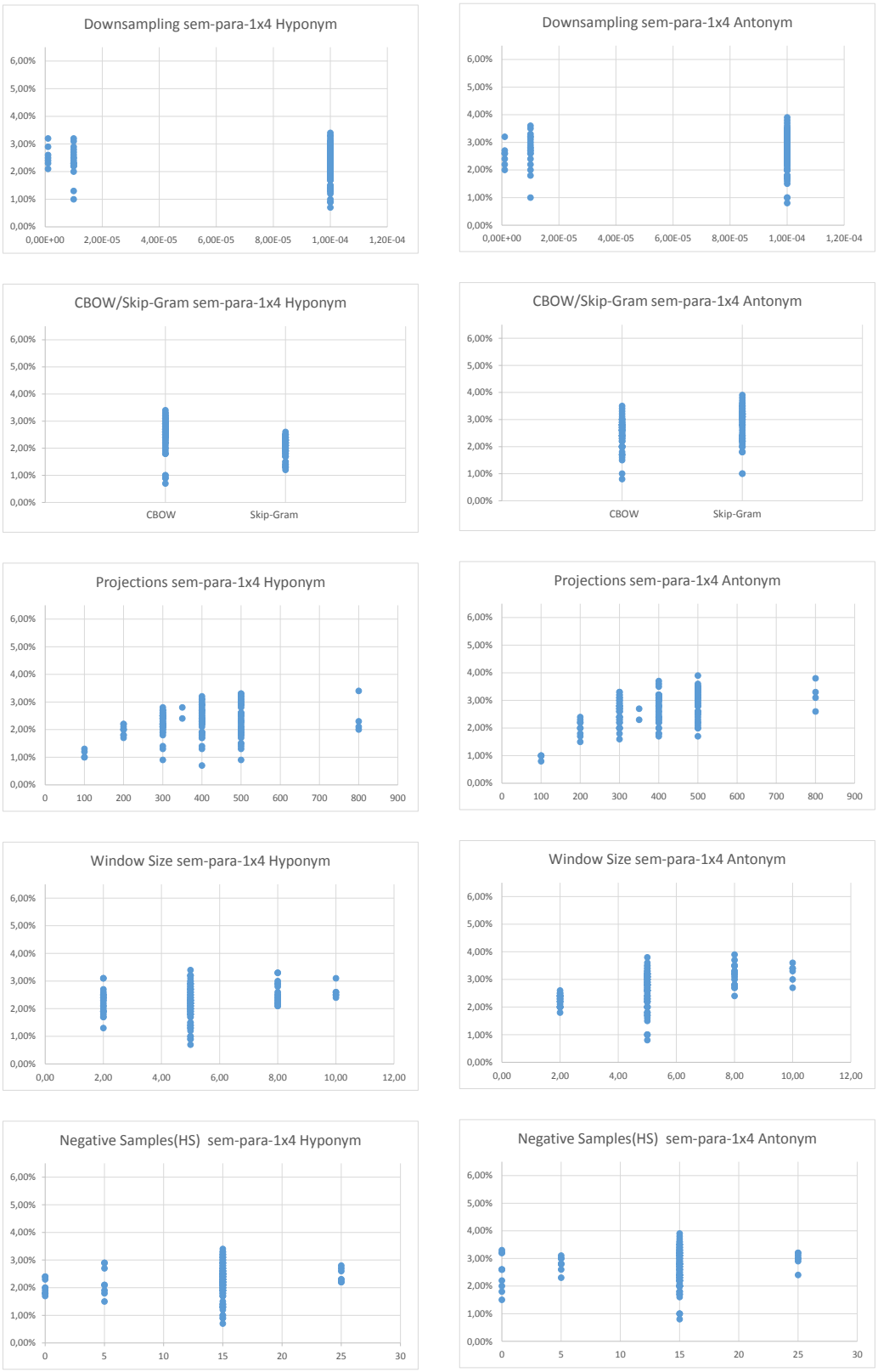


Figure 8.7: Accuracy for *sem-para-1x4* Test Set (1). *Each model is represented with a score of how many times overall the second analogy word pair was complemented correctly, computed with 3CosMul. Scores are presented separately for hyponym, synonym, antonym task sections and in total.*

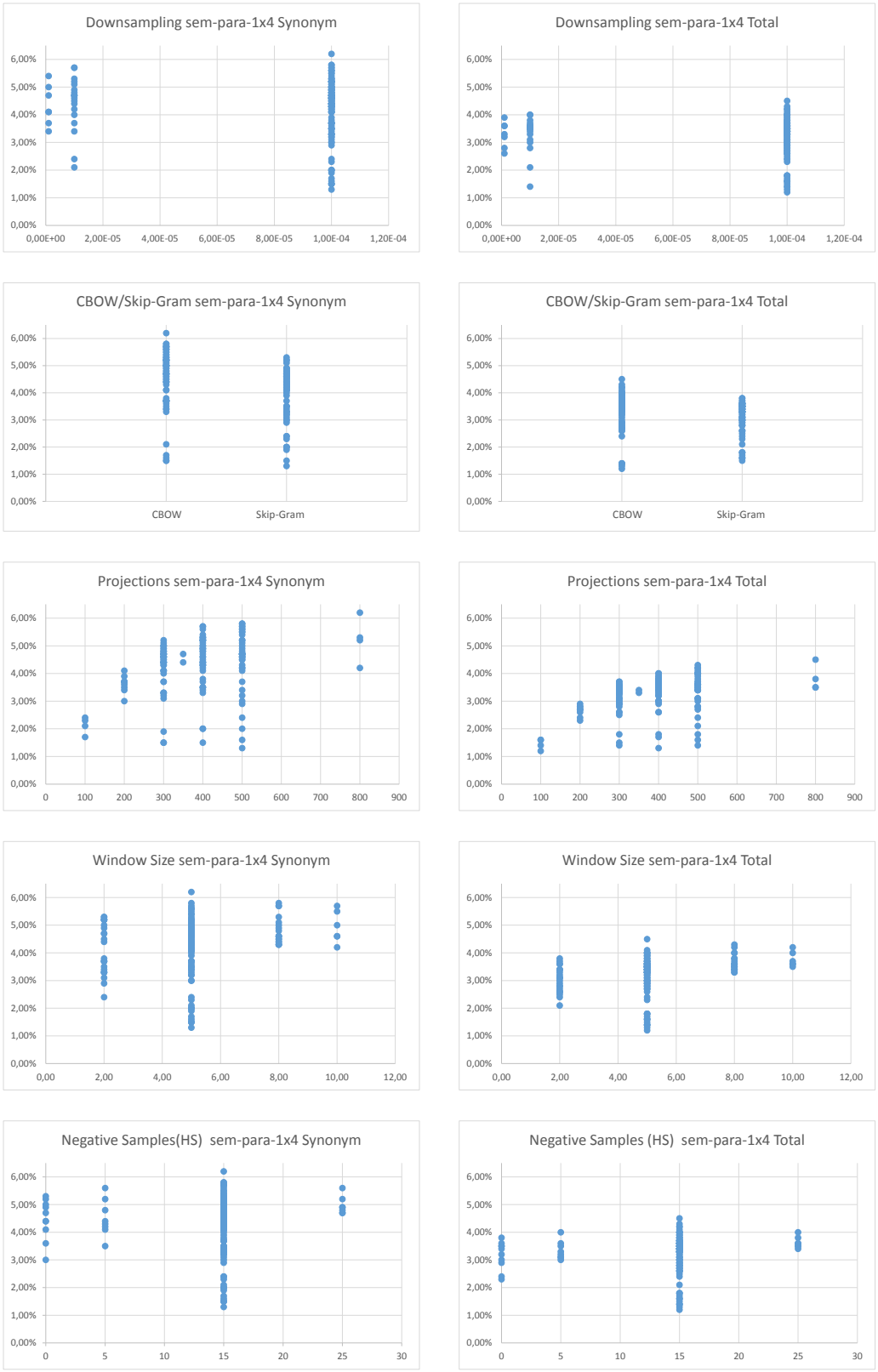
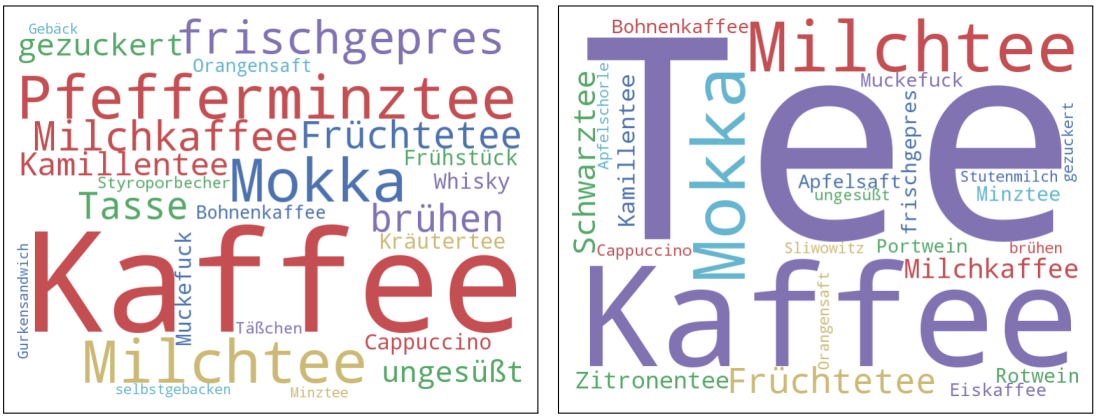


Figure 8.8: Accuracy for *sem-para-1x4* Test Set (2). *Each model is represented with a score of how many times overall the second analogy word pair was complemented correctly, computed with 3CosMul. Scores are presented separately for hyponym, synonym, antonym task sections and in total.*

C Clustering Evaluations

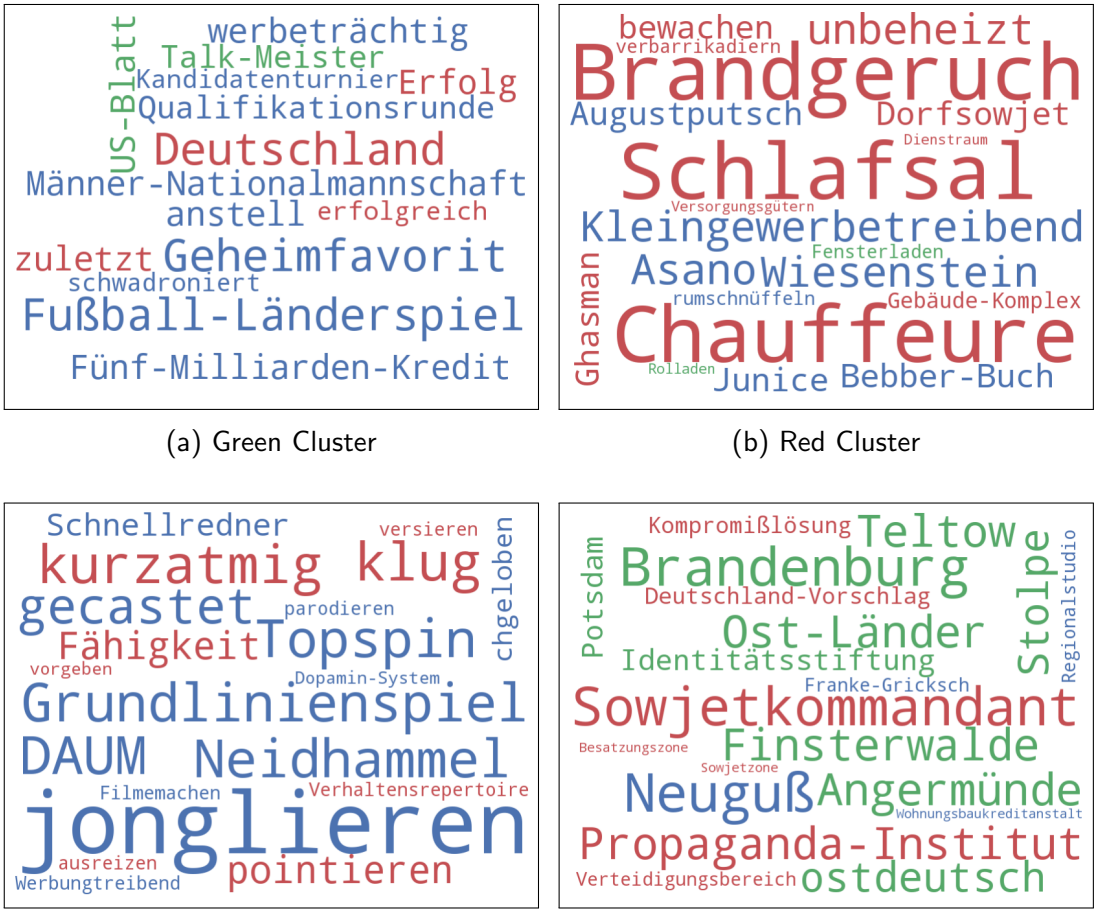
The following constitutes a supplement to clusterings discussed in chapter 6.



(a) *Tee* (tea) (b) *Pfefferminztee* (peppermint tea)

Figure 8.9: Frequency-Based Word Cloud for Local Taxonomy *Getränk* (Beverage). *Computed with* LemmaC.

Figure 8.10: Frequency-Based Word Cloud *Tor* (gate/goal). *Computed with* LemmaC.



(a) Green Cluster

(b) Red Cluster

(c) Purple Cluster

(d) Blue Cluster

Figure 8.11: Renewed Search and Clustering Based on Positive and Negative Similarities. *Computed with* LemmaC. *Both the positive and negative terms were used as the starting point to a new search. The results show this leads to heterogeneous results with little or no relation to the positive cluster.*

(a) Green Cluster

(b) Red Cluster

(c) Purple Cluster

(d) Blue Cluster

Figure 8.12: Renewed Search and Clustering Based on Positive Similarities. *Computed with* LemmaC. *Members of the respective cluster were used as the starting point to a new search.*